2013-09

Efficient strategies for active interface-level network topology discovery

Baltra, Guillermo P.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

EFFICIENT STRATEGIES FOR ACTIVE INTERFACE-LEVEL NETWORK TOPOLOGY DISCOVERY

by

Guillermo P. Baltra

September 2013

Thesis Advisors: Preetha Thulasiraman
Robert Beverly

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
18–9–2013	Master's Thesis	2011-09-26—2013-09-27

4. TITLE AND SUBTITLE

EFFICIENT STRATEGIES FOR ACTIVE INTERFACE-LEVEL NETWORK TOPOLOGY DISCOVERY

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Guillermo P. Baltra

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.IRB Protocol Number: XXXX

14. ABSTRACT

As a piece of critical infrastructure, the Internet brings both benefits and security concerns. Recent cyber-security episodes such as route hijacks and Denial-of-Service attacks might have been mitigated and prevented with better knowledge of the network's logical topology; i.e., router nodes and links. Current production public active mapping systems; e.g., Ark, Rocketfuel, and iPlane, produce valuable inferences of the Internet's topology, as well as facilitating longitudinal analysis. We examine the extent to which the techniques utilized by these existing systems can be improved, in particular by attempting to reduce their high probing load. Our methodology divides the discovery process into three steps: destination selection, monitor assignment, and stop criterion. We implement and evaluate alternative designs for each step. The complete system runs in real-time on a production system to probe 500 randomly selected Internet subnetworks and gather real-world network maps. As compared to datasets from existing measurement platforms, we find that our method is able to generate 80% of the amount of data with 69% less load.

15. SUBJECT TERMS

Internet Topology, Network Topology, Adaptive Probing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	99	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

EFFICIENT STRATEGIES FOR ACTIVE INTERFACE-LEVEL NETWORK TOPOLOGY DISCOVERY

Guillermo P. Baltra
Lieutenant, Chile Navy
B.S., Academia Politécnica Naval, Viña del Mar, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author: Guillermo P. Baltra

Approved by: Dr. Preetha Thulasiraman
Thesis Advisor

Dr. Robert Beverly
Thesis Advisor

Dr. Clark Robertson
Chair, Department of Electrical and Computer Engineering

Dr. Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As a piece of critical infrastructure, the Internet brings both benefits and security concerns. Recent cyber-security episodes such as route hijacks and Denial-of-Service attacks might have been mitigated and prevented with better knowledge of the network's logical topology; i.e., router nodes and links. Current production public active mapping systems; e.g., Ark, Rocketfuel, and iPlane, produce valuable inferences of the Internet's topology, as well as facilitating longitudinal analysis. We examine the extent to which the techniques utilized by these existing systems can be improved, in particular by attempting to reduce their high probing load. Our methodology divides the discovery process into three steps: destination selection, monitor assignment, and stop criterion. We implement and evaluate alternative designs for each step. The complete system runs in real-time on a production system to probe 500 randomly selected Internet subnetworks and gather real-world network maps. As compared to datasets from existing measurement platforms, we find that our method is able to generate 80% of the amount of data with 69% less load.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AfriNIC	African Network Information Centre
API	Application Programming Interface
APNIC	Asia-Pacific Network Information Centre
ARIN	American Registry for Internet Numbers
Ark	Archipelago
AS	Autonomous System
ASN	Autonomous System Number
BGP	Border Gateway Protocol
CAIDA	Cooperative Association for Internet Data Analysis
CERT	Computer Emergency Response Team
CIDR	Classless Inter-Domain Routing
DGSSM	Delay-Guiding Source Selection Method
DIMES	Distributed Internet MEasurements and Simulations
DoD	Department of Defense
DoS	Denial-of-Service
DNS	Domain Name System
ERP	Exterior Router Protocol
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPS	Ingress Point Spreading

IRP	Interior Router Protocol
ISC	Interface Set Cover
ISP	Internet Service Provider
LACNIC	Latin America and Caribbean Network Information Centre
LCP	Least Common Prefix
LD	Levenshtein Distance
MAX	Maximum Coverage
NAT	Network Address Translation
NID	New Interface Discovery
NRO	Number Resource Organization
OSI	Open Systems Interconnection
RIPE NCC	Réseaux IP Européens Network Coordination Centre
RIR	Regional Internet Registry
RIS	Routing Information Service
SCP	Subnet Centric Probing
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
SDIS	Logic Distance
VoIP	Voice over Internet Protocol
VP	Vantage Point
VPS	Vantage Point Spreading

Executive Summary

Since the mid-1990s, the Internet has had a revolutionary impact on culture, commerce and day-to-day activities. As a piece of critical infrastructure, the Internet brings both benefits and security concerns. Recent cyber-security episodes such as route hijacks and Denial-of-Service attacks might have been better mitigated and prevented with better knowledge of the network's logical topology, i.e., constituent routers and router links. Current production public active mapping systems, e.g., Ark, Rocketfuel, and iPlane, produce valuable inferences of the Internet's topology, as well as facilitating longitudinal analysis. We examine the extent to which the techniques utilized by these existing systems can be improved, in particular by attempting to reduce their high probing load. Reducing probing load has many practical benefits and can potentially lead to improved network maps and a better ability to capture transient network dynamics.

This thesis builds on a recently proposed technique in the academic literature, Subnet Centric Probing (SCP). SCP is an active probing strategy that seeks to send enough probes to capture the internal structure of a target subnetwork while avoiding probes that contribute no additional mapping value. SCP thus attempts to decrease probing load while producing reliable network topology maps. Although SCP has been shown to work well in simulation over real-data from CAIDA's Archipelago platform, SCP has not previously been implemented or tested in production.

The first contribution of this thesis is the implementation and comprehensive testing of SCP to perform real, production, Internet mapping. We find that, as originally proposed, SCP has a critical flaw stemming load-balancing in the Internet, which is now common practice. In particular, SCP's use of the Levenshtein Distance (LD) metric as a stopping criterion is disrupted by load balancing, thereby causing the SCP algorithm to probe every address within the target subnetwork, defeating the intent of SCP.

Our second contribution is a series of improvements to address our real-world experience in implementing and deploying SCP. We propose to divide the discovery process into three steps: destination selection, monitor assignment, and stop criterion. This layered structure allows us to analyze each step of the process independently.

We take globally visible Border Gateway Protocol (BGP) network prefixes as the input to our probing. These prefixes are derived from the public Routeviews looking-glass server. To select

destinations to probe within each prefix, we make use of the Least Common Prefix (LCP) algorithm, taken from the original SCP algorithm. LCP divides prefix ranges into maximally distant sub-networks, where the results of probing each portion of the parent prefix drive subsequent probing steps.

Next, we contribute and analyze a new stopping criterion, New Interface Discovery (NID), to replace LD. In NID, rather than comparing complete paths, only interface hops that are observed during probing that belong to the destination Autonomous System (AS) are considered. Further, while SCP's use of LD was pair-wise and memoryless, NID compares traces to the set of previously discovered interfaces within the destination AS. Pair-wise independence allows prefixes to be probed from various monitors with an unrestricted number of traces.

Previous work has examined differences between basic monitor assignment strategies, including random with and without replacement. We introduce two new monitor assignment techniques: Maximum Coverage (MAX) and Ingress Point Spreading (IPS). Both these techniques analyze previous rounds of probing to more intelligently assign monitors to the set of destination addresses SCP probes. The high-level purpose of these techniques is to discover as many diverse paths into the destination network, thereby avoiding early termination by SCP. In particular, MAX's objective is to find as many nodes and links outside the destination AS as possible and, thus, obtain a richer global topology map; discovering ingress points to destination prefixes is a side effect. MAX uses /8 prefixes as "logical landmarks" since they are typically physically bounded by the region to which each Regional Internet Registry (RIR) is assigned. Monitors are rank ordered based on hop distance and path diversity to the landmark. MAX thus attempts to maximize the coverage of each trace from source to destination given prior knowledge.

In contrast to MAX, IPS explicitly aims to discover ingress routers to target networks, which in turn leads to path diversity. By again screening traces obtained from previous rounds of probing, IPS finds ingress routers to the trace's destination address /8 prefix. By targeting known ingress routers, IPS attempts to increase the chances of subsequent probes following similar routes and traversing different ingress points. Finally, we examine a variation on IPS that explores the inherent trade off between probing load and coverage. IPS++ sends one probe per ingress router to the original prefix in order to encourage traversal of all known possible ingresses.

As shown in Figure 1, the random method, Vantage Point Spreading (VPS), MAX and IPS each discover over 57% of the number of vertices as compared to the existing Ark system with less than 20% of the probing load. The best performance is given by IPS with 58% of vertices and

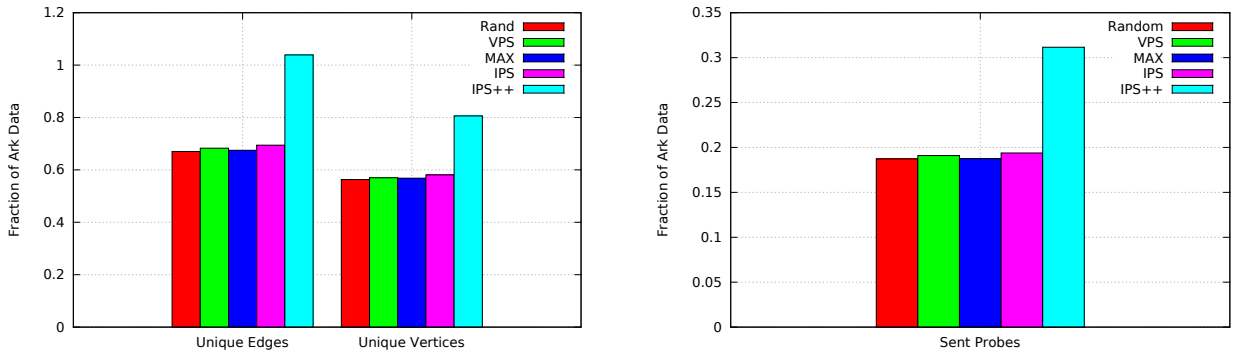

Figure 1. Results comparison between different methods.

over 69% of edges. Further, IPS sent more probes before finishing, which we observe to be directly correlated to the method's general performance and, thus, suggests that rank ordering the list of vantage points avoids early termination. By increasing the load to just 31%, IPS++ results in 80% of the number of vertices discovered by Ark and more edges than discovered by Ark. These empirical results suggest that there is practical utility in the topology mapping algorithms we developed.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

Foremost, I would like to express my sincere gratitude to both my advisors. To Prof. Beverly for continuous encouragement and advice during my research. His guidance helped me in all the time of research and writing of this thesis. To Prof. Thulasiraman for her patience and enthusiasm for my ideas and for promptly helping me when needed.

Besides my advisors, I would like to thank Prof. Xie and Prof. Ralucca for being actively involved in my thesis research and for their comments and suggestions.

Last, but not least, I would like to thank my wife, Gabriela, for accompanying me during these two years full of new challenges and sacrifices and for giving me one of my greatest joys in life, Antonia.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

The telegraph, the Atlantic cable, the telephone, the cellular phone, and more recently the Internet; every single one of these communication technologies changed the world in its time. All of them made their way into people's daily lives, making it hard to imagine how life went on before their introduction. Since the mid-1990s, the Internet has had a revolutionary impact on culture and commerce, starting with the rise of electronic mail whichi, is expected to have 3.8 billion user accounts by 2014 [1]. The growth of instant messaging and social networking accounts are predicted to be over 3.5 and 3.7 billion, respectively, by 2014. The growing influence of the Internet has moved many researchers towards understanding how it operates. As with past communication technologies, the Internet not only brings benefits but also many concerns. In particular, both governments and private organizations must balance the benefits of the Internet with its security issues. Recent cyber-security episodes such as route hijacks and Denial-of-Service (DoS) attacks could have been mitigated by better knowledge of the network's logical topology. We can reasonably ask ourselves: what is the topology of the Internet? Sometimes referred to as a network of networks, the Internet is a highly complex and distributed interconnection of computing systems that continuously evolves. This cyberspace network is difficult to observe because its original design does not facilitate taking adequate measurements of network performance. In addition, organizations actively hide information from public access, further complicating researchers' undertakings. Thus, the Internet remains poorly understood.

1.1 Motivation

The purpose of this thesis is to analyze the Internet, in particular its topology at an interface-level, using active measurement strategies. Network topology is a field of networking that has received increasing interest from the research community. Researchers have devoted considerable effort to comprehend its nature, but there is still space for growth. Each small step taken towards its understanding might have an impact not only inside the research community but to Internet consumers at large. In relation to research, it is important for proper network modeling to have a known topology to work with. The validation of improvements to current protocols and new architectures must be accomplished through a known topology. Rigorous testing is needed in order to verify backwards compatibility with currently deployed standards and also to make sure that there are no disruptions that might cause the whole network to collapse during

the implementation. By comparing corresponding Internet topologies in different time periods, we can understand the network's growth and even make predictions about its evolutionary direction.

From the security point of view, knowledge and comprehension of network topology provides several benefits to organizations such as Internet Watch Centers and Computer Emergency Response Teams (CERTs) for anomaly detection due to malicious attacks, misconfigurations, or faults, and network troubleshooting and diagnosis. In this regard, network robustness is a main concern, specifically in terms of how the network deals with failures and attacks and how the network recovers from them. Similarly, knowledge of an opponent's network topology grants an opportunity for mounting varied kinds of attacks, including DoS, link-removal, and route hijack.

Current topology discovery tools do not satisfy the need of having fast data gathering, which leads to missing events of interest that might happen during the whole mapping process. For example, the Cooperative Association for Internet Data Analysis (CAIDA) runs a project that implements an active measurement infrastructure called Archipelago (Ark) [2]. To complete a full cycle of information gathering on the Internet, which amounts to 9.5 million probes, takes approximately three days. As said earlier, many events of interest that can occur on the Internet are small in nature. For example, just one Border Gateway Protocol (BGP) update is enough for a route hijack. On February 24, 2008, the Pakistani Telecom, in an attempt to block YouTube access within their country, took down YouTube entirely from the whole Internet.On April 8, 2010, China Telecom originated 37,000 prefixes not belonging to them in 15 minutes, causing massive outage of services globally. There are other similar projects at Ark that are dedicated to Internet topology research, including Dolphin [3], iPlane [4], Distributed Internet MEasurements and Simulations (DIMES) [5] and Rocketfuel [6]. These arefurther analyzed in Chapter 3.

As mentioned before, having knowledge of the topology of a network allows the discovery of vulnerabilities and bottlenecks. Reconnaissance is a term used in networking to describe the gathering of information by an unauthorized agent in order to get the mapping of systems, services, or vulnerabilities. It usually precedes in most cases an actual access to the system or a DoS attack. A DoS attack is the prevention of authorized users from using a specific service by exhausting the system's resources. Several examples can be recalled. Some of the most notorious attacks happened during the year 2000 on very well known victims such as

eBay, Amazon, Buy.com and Yahoo. DoS attacks and, specifically, its distributed form are an unsolved issue.

1.2 Topology Mapping Challenges

A very distinctive characteristic of the Internet is its distributed structure with no centralized administration. This very important feature adds to the scalability of the Internet and allows it to grow organically. The Internet, as it was originally conceived, has no measurement implementations techniques. In addition, it has a dynamic topology making no two Internet graphs obtained during different periods of time equivalent.

Due to the lack of a ground truth, current topology inference techniques are fragile, leaving large amounts of space for assumptions. In that sense, the Internet can still be even more elusive to researchers trying to map it, since it naturally hides information for scaling considerations, such as in BGP and for economic reasons by commercial Internet Service Providers (ISPs).

Anonymous routers appear very often while probing the global network. They are commonly used because anonymous routers are less likely to be targeted by malicious counterparts and also because these routers allow ISPs to keep their topologies opaque. As was found in [7], of the 18 million traces sent, nine million anonymous routers were found. These results show that if anonymous routers are not properly dealt with, they might introduce significant errors in connectivity, accuracy and resolution in the inferred topology.

There are other issues that present themselves as obstacles during active probing that are worth mentioning and where current inference techniques lack efficient solutions, such as link failures between Autonomous Systems (ASes) or within an AS, and application-level failures produced by server crashes. Also, Network Address Translations (NATs) and firewalls introduce the middle-box problem. Active probing, which is discussed in Chapter 3, has serious difficulties identifying routers protected by NATs or firewalls. Finally, due to the fact that the Internet is a dynamic network, temporal links, which are links that only operate under certain circumstances, are rarely discovered and only when those conditions get triggered.

Related academic research has proposed diverse schemes for decreasing the time required for active mapping while producing reliable topology graphs. In [8] the authors propose an algorithm to benefit from the tree-like structure of traces from one source towards different destinations. Cheleby [9] is a probing system that reduces network burden by finding a full trace towards a destination inside an AS and then modifying the Time-To-Live (TTL) value to start

probing from the ingress router which is the last hop that does not belong to the AS. In [10] the authors introduce a new metric oriented to monitor assignment that uses the underlying link delay between the source and a small number of landmarks.

1.3 Goals and Objectives

The broad objective of this thesis is to provide accurate network graphs at an interface granularity even at a very large scale such as the Internet. This is done by following the research started in [11], continuing with its improvement, addressing its weaknesses and taking the step of implementing it on the Internet. This study focuses on combining the tools presented in the mentioned research and comparing results with what an equivalent production measurement system would generate. This means that all the information presented is gathered from the Internet, and every experiment is conducted on that same global network.

By the use of primitives, it was shown in [11] that in comparison with current production tools, such as Ark, it is possible to increase the frequency at which mapping of the network is executed and at the same time, to decrease the load on the network generated by the probing itself. The three proposed primitives are called Subnet Centric Probing (SCP), Interface Set Cover (ISC) and Vantage Point Spreading (VPS). All three have shown to be effective in providing the desired results on their own, but still there is work to be done in terms of an algorithm that allows the primitives to work as a combined effort.

The main question that this work addresses is how SCP and VPS can be integrated so that it is possible to decrease the size of a set of destinations while using as many distinct Vantage Points (VPs) as possible to probe within a given BGP prefix. The latter is validated following an iterative methodology over a previously defined ground truth. Behind the research effort lies the intention of obtaining network topologies an order of magnitude faster than existing systems in order to capture transient dynamics, including malicious or misconfiguration events.

1.4 Thesis Contributions

In this thesis, an effective strategy for extracting ASes internal structures is designed. The main goal is to implement the algorithm on the Internet through a productive measurement tool. The strategy extracts available AS-level topology, which it uses to direct its probing to obtain the AS internal structure at an interface-level topology.

So far a similar strategy, the one from which this thesis inherits concepts and terminology, has

been successfully simulated over a previously probed data set [11]. The main contribution of this work is to be able to implement that probing strategy in real-time on the Internet.

New methodologies for efficient VP assignments are evaluated. The first one, named the Maximum Coverage (MAX) method, examines the plausibility of monitor selection according to their logical location on the network. By calculating the relative distance of VPs in relation to a network and the feasible paths followed by probes toward that same network, it is possible to maximize node and link discovery by sent probes.

The second monitor-assignment technique, called Ingress Point Spreading (IPS), aims at finding ingress routers to selected destination networks. By discovering new entrances into the networks, we can capture more thoroughly networks' internal structure.

1.5 Thesis Organization

Some background for understanding the nature of statements and contributions made in this thesis, with a description of Internet routing policies, topology, administrative structure and available information sources for topology discovery, is provided in Chapter 2. Related work regarding state of the art mapping techniques, the network-structure oriented SCP algorithm, and their respective limitations are described in Chapter 3. The designs of the proposed solutions and further improvements, as well as the implementations details, are explained in Chapter 4. The objectives delineated by the thesis and analyzes the results provided by the proposed solutions, are discussed in Chapter 5. The results and contributions granted by the thesis and gives further guidelines about future research are given in Chapter 6. The code used to implement the methods described in this thesis are shown in the Appendix.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

An introduction to some Internet elements involved in network mapping, such as its basic terminology, relevant routing protocols, and the different topology levels and their characteristics is provided in this chapter. Additionally, we also present some of the available tools currently employed in network topology discovery.

To start the discussion on topology mapping of a complex network such as the Internet, it is convenient to begin by understanding one of its main components, the AS. An AS is defined as a group of networks that use their own independent routing protocol and is managed by a single organization. The Internet is composed of tens of thousands of loosely connected ASes. Each AS is identified by a unique 32 bit number called the Autonomous System Number (ASN). ASNs are assigned in blocks by the Internet Assigned Numbers Authority (IANA) to the corresponding Regional Internet Registry (RIR). The RIR then assigns ASNs to entities within its designated area of responsibility from the assigned number range by IANA. More about RIRs is discussed in Section 2.3.

There have been several attempts to measure the Internet topology graph at a variety of granularities. It is possible to distinguish two different measurement methodologies for topology discovery employed by researchers: the *active* approach and the *passive* approach. Active methods rely on the injection of measurement probes such as pings and traceroutes, and the projects that have implemented this approach are thoroughly discussed in Section 3.2. Passive methods execute a non-intrusive observation of the network by analyzing BGP route announcements. In the research literature, most work on passive BGP analysis has focused on publicly available default-free BGP vantage points, including Routeviews [12] and Routing Information Service (RIS) of Réseaux IP Européens Network Coordination Centre (RIPE NCC) [13]).

2.1 Routing protocols

A routing protocol is a protocol related to the network layer and allows routers to exchange information with each other in order to permit the correct routing of packets to the proper destination. The use of these type of protocols for the construction of the dynamic routing tables is necessary when the number of interconnected subnetworks is high, as it is the case of the Internet.

Given the large number networks attached to the Internet, when it comes to routing it is convenient to see the network as a set of ASes, each of which handles independently and uniformly its internal routing. The protocols vary depending on whether the involved routers connect to each other within the same AS, called an Interior Router Protocol (IRP), or interconnect different ASes, called an Exterior Router Protocol (ERP). The independence between IRP and ERP gives flexibility for tailor made protocols on the inside of each AS.

2.1.1 The Internet Protocol

Internet Protocol (IP) is the network protocol part of the IP suite (Transmission Control Protocol (TCP)/IP) on which the operation of the Internet is based. It belongs to layer three of the Open Systems Interconnection (OSI) model and was created to interconnect heterogeneous networks in terms of technology, performance and management. It is implemented over other protocols at the link layer such as the Ethernet protocol. It is a connectionless protocol of the best effort type, which means it does not guarantee any form of communication reliability in relation to error control, flow control and congestion control. Therefore, the latter has to be compensated for at the transport layer with protocols such as TCP. The currently used version of IP is also known as Internet Protocol version 4 (IPv4) to distinguish it from the most recent Internet Protocol version 6 (IPv6). IPv6 was developed because of the need to satisfy the increasing demand of IP addresses due to the large number of computers connecting to the Internet. Particularly, IPv4 addresses were standardized as a 32-bit number in 1980 by the Department of Defense (DoD) [14], and the pool of available numbers has been depleted at a rate not anticipated in its original design and is now near to the address space exhaustion. While this thesis explores only IPv4 topology mapping, the techniques herein may also be useful to IPv6 in future work.

2.1.2 IP address prefixes

A subnetwork, or *subnet*, is a logical subdivision of an IP network and typically corresponds to an underlying local network. The action of *subnetting* refers to the division of a network into two or more parts. A subnet mask is used to distinguish the part of the network's address range used for routing and for hosts' address assignments.

A network's routing *prefix* corresponds to the most significant bits of an IP address. It precedes the part of the address used as the host's identifier. The routing prefixes are expressed in the Classless Inter-Domain Routing (CIDR) notation, which uses the first address of a network followed by the number of bits used by the prefix, separated by the slash character '/'. For example, if 192.168.1.0/24 is the network's prefix, it means that the indicated IPv4 address is

the starting address, and that 24 bits are allocated for the network's number. The eight bits left indicate the number of available host addresses. In other words, there are 254 available host addresses without considering the network and broadcasting addresses.

2.1.3 Border Gateway Protocol

The protocol used to make core routing decisions on the Internet by exchanging reachability information among ASes is known as BGP [15]. This is a protocol that allows the Internet to be fully decentralized and version four is the one in use since 1994. BGP routing supports CIDR and uses route aggregation to decrease the number of prefixes appearing in the global routing table. BGP network information exchange is done by establishing a communication session between edge routers in autonomous ASes based on TCP. This session stays connected and allows periodical exchanges of information and route updates between both ends of the communication.

2.2 Internet Topology

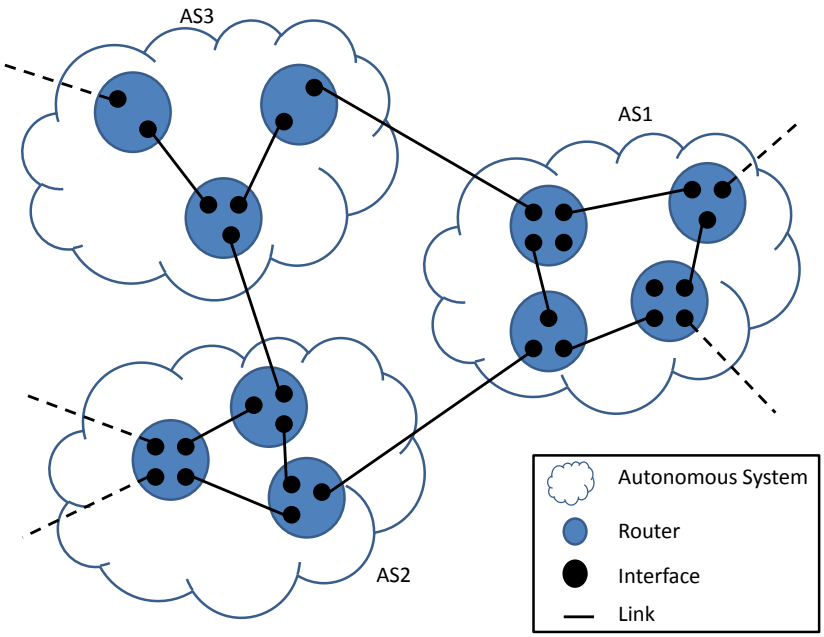

Figure 2.1. Different levels of granularity of the Internet topology.

In terms of *Internet topology*, even though there are other levels of granularity considered by researchers, there are three that are the most commonly used: AS-level, router-level and interface-level. From these three levels, the first and last are discussed, since they are directly related to

the subject at hand, in particular the interface-level topology. In a router-level topology, nodes represent routers and links indicate one-hop connectivity between routers. In Figure 2.1 the different levels of granularity of the Internet topology can be seen, where the large white areas represent the AS-level topology, the small blue areas represent each router and, therefore, router-level topology, and finally, the black dots represent the interface-level topology.

2.2.1 AS-level Topology

The entire Internet is usually considered as an AS-level topology graph where each AS is a node, and the BGP peering between two ASes is an edge. For some time the AS-level topology has been gathering interest from political, economic and academic players due to its relevance in day-to-day Internet operations and research, which include activities such as AS-level topology inference and AS relationships [16] [17] [18] and network topology generators [19] [20] [21]. Researchers have made several attempts on mapping the Internet at this level of topology using passive and active methodologies. Active methods are very similar to those used at the interface level topology and are covered in Chapter 3. On the other hand, common sources of information used in passive strategies are the Routeviews project, RIS of RIPE NCC and the CIDR-report.

Routeviews [12] is a project managed by the University of Oregon and consists of providing real-time information about the global routing system from different perspectives and locations. Currently, it has distributed around the globe 16 routers with over 500 peering sessions in total. Basically, what Routeviews does is to collect, without providing transit services, BGP updates coming from these peers. Routeviews databases have been used to infer AS peering relationships [22], to map AS-level topology [18], to detect routing anomalies in a network [23] and to visualize BGP routing changes [24], among others.

RIS [13] is a RIPE NCC project oriented to the collection and storage of Internet routing data from several locations around the globe through the use of routing beacons. RIS offers tools such as notifications on rogue BGP announcements, knowledge of whether an ASN is in use and statistical data about its neighbors and knowledge of when a prefix was last seen on the Internet.

The *CIDR-report* [25] was originally established to quantify and track the growth rate of global routing states. Its methodology consists of analyzing the BGP table within AS2.0 and generating an aggregation report for each individual AS. The report provides for each AS a list of its adjacent ASes and its announced prefix. The CIDR-report has been used as a topology graphing

tool in projects such as [26].

2.2.2 Interface-level Topology

The interface-level topology is the most detailed level of the Internet Topology. This level describes the interfaces of routers and end hosts. In a topology-graph, an interface is represented by a node, and each edge illustrates the link-layer connection between nodes. To know which interfaces belong to the same router, or in other words, to cross from an interface-level to a router-level topology, a technique called alias resolution can be performed. Since this technique is out of the scope of this study, more information about it can be found in [27].

In [28] the authors try to determine topology mapping algorithms to discover both intra-domain and Internet backbone topology. The requirements established for each algorithm are efficiency, speed, completeness and accuracy. With those requirements in mind, four different strategies were specified: Simple Network Management Protocol (SNMP), Domain Name System (DNS) zone transfer with broadcast ping, DNS zone transfer with traceroute and probing with traceroute. The first three algorithms are less applicable due to the lack of reliability since they need specific features that are less probable to be enabled for security reasons. On the other hand, the benefits of using traceroute as a topology discovery tool are significant, the first of them being that it provides the complete paths followed by probes.

2.2.3 Traceroute

Traceroute is a hop-limited probing tool that attempts to discover the path data packets follow. Initially deployed as an experimental tool for network diagnosis, now traceroute is widely used across the globe for diverse purposes. This tool sends TCP, User Datagram Protocol (UDP) or Internet Control Message Protocol (ICMP) packets to the destination address, dynamically increasing their TTL values. The TTL value is the hop limit count before the packet gets discarded by a router. This means that every time the packet passes through an intermediate router the TTL value is decremented by one until it reaches zero and is discarded. The router that discards the packet sends an ICMP error message *time exceeded*. This error message provides the IP address from an interface in the router that discarded the packet and, therefore, the address of an intermediate node. Although theoretically the router could respond from any of its addresses, in practice, the response is often routed back through the same interface [27].

The way traceroute works, as it can be seen in Figure 2.2, is by gradually increasing the TTL value of sent packets and starting with a value equal to one until reaching the destination which

ends the procedure by sending ICMP messages *echo reply* when an ICMP packet was initially sent and *port unreachable* for UDP and TCP packets.

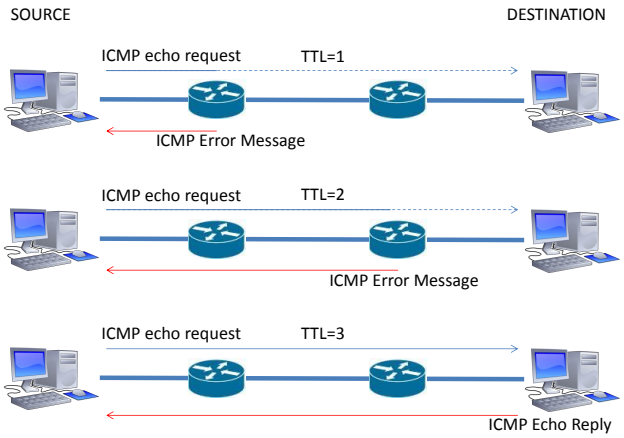

Figure 2.2. Traceroute sending ICMP echo request, error and receiving echo reply messages.

Unfortunately, the latter case is the ideal situation. When implementing this tool on the Internet, load balancing routers cause it to not work as expected. In [29] the authors explored three different types of load balancing policies used by routers: *per-packet*, *per-flow* and *per-destination* load balancing. In terms of measurement, per-destination load balancing resembles classic routing. Per-packet load balancing focuses on evenly distributing data traffic across the available paths. In per-flow load balancing, the router forwards all packets belonging to the same flow through the same interface. A flow is determined by analyzing packet headers. A five-tuple composed of source IP address, destination IP address, protocol, and both TCP or UDP source and destination ports is used as the flow identifier.

Figure 2.3 (a) is an example of what a network with a load balancer router might look like. In Figure 2.3 (b) through (e) are illustrations of how traceroute performs in this kind of environment. Every time a new packet is sent, the TTL value is increased by one, but since the load balancer distributes the loads into different interfaces, packets end up following different paths. One possible resulting path by traceroute can be seen in Figure 2.3 (f), where a false link distorts the inferred topology.

Load balancing is very frequently present on the Internet. In an experiment conducted in [30] to measure the occurrences of load-balancing, 91% of the studied traces were affected by some

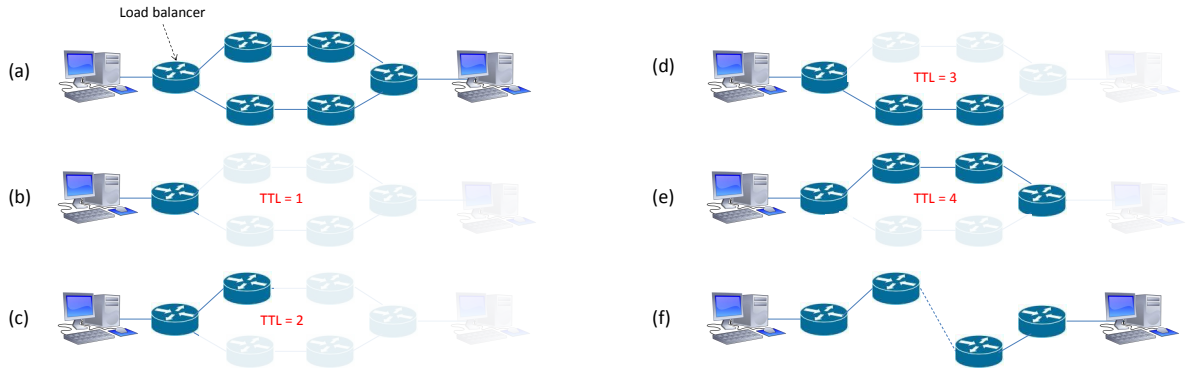

Figure 2.3. Influence of load balancing over traceroute.

kind of load balancing. Out of the same data set, 72% showed per-destination load balancing, 39% showed per-flow load balancing, and only 2% showed per packet load balancing.

As explained in [29] traceroute's own algorithm causes unwanted behavior where there is per-flow load balancing. Figure 2.4 is an illustration of the different types of headers and the fields that get changed by traceroute. As a product of this change, the five-tuple varies and per-flow load balancing routers forward packets to different interfaces even when they belong to the same flow. That is why the authors of the research in [29] introduced a new kind of tool which they named *paris-traceroute*. The main idea behind paris-traceroute is to keep state in those fields required to stay in the same flow. This is done by manipulating the packet's load to vary the Checksum field in UDP packets or by changing the identifier to keep the Checksum constant in ICMP packets. As for TCP packets, the only manipulation needed is to vary the Sequence Number field.

Traceroute provides raw paths which may contain irregularities due to factors such as non-responsive hops [7], loops, cycles and diamonds [29]. A non-responsive hop appears in a path when the router discarding the traceroute packet does not generate an error message, resulting in a gap between two known IP addresses. The gap is usually represented by an asterisk "*". The research in [7] analyzes different strategies to accomplish unresponsive router resolution. A loop is generated when the same address repeats itself in the same path. This can be possibly caused by load-balancing, routing changes, misconfigured routers that forward packets with TTL equal to zero and address rewriting by gateway routers such as NAT boxes. The resulting topologies depend on how these anomalies are handled.

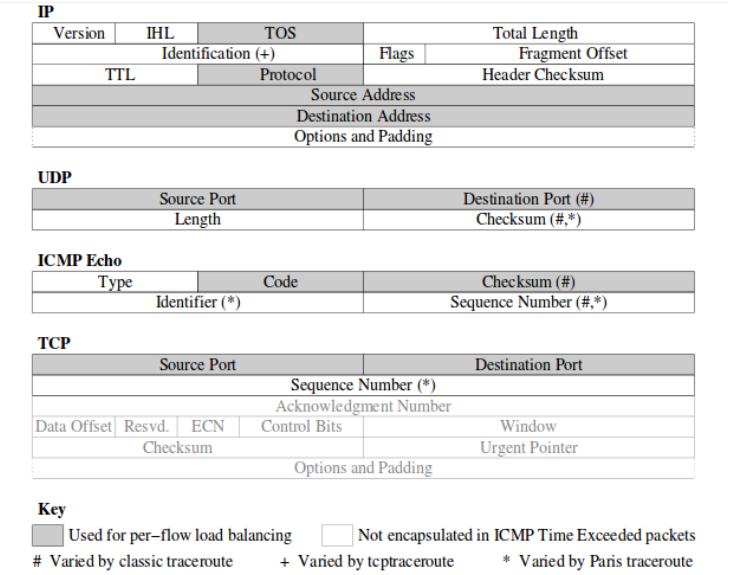

Figure 2.4. Packet header fields used modified by the different probing techniques [29].

When probing the Internet, traceroute probes might produce incomplete paths. This happens when the last hops in the path come in the form of a *Request timed out* error message. There are several reasons why probes do not complete a trace. Firewalls filtering certain ports or interfaces hiding behind NAT boxes might be blocking traceroutes. The return route might differ from the forward route, which might present a problem. There also may be a connection problem at that particular router or with the following one.

2.3 Regional Internet Registries

As was specified during its standardization in 1980 by the DoD [14], IPv4 addresses are designed as a 32-bit number. Naturally, the address space being fixed, this limits the total number of available addresses for assignment. Therefore, it was necessary for some organizations to be in charge of allocating this limited budget of addresses in predetermined areas of responsibility; thus, the RIRs were created. An RIR is the organization in charge the allocation and registration of IP addresses and AS numbers. Each RIR has been assigned IPv4 addresses in terms of /8 prefixes; this allocation is controlled by the IANA, whose guidelines can be verified in [31].

Initially, three RIRs were formed: American Registry for Internet Numbers (ARIN), RIPE NCC and Asia-Pacific Network Information Centre (APNIC). In 2001, the Internet Corporation for Assigned Names and Numbers (ICANN) published the basis for the establishment of new RIRs [32]; thus, two new RIRs were formed: the Latin America and Caribbean Network In-

formation Centre (LACNIC) in 2002 and African Network Information Centre (AfriNIC) in 2005. Currently, the RIRs' areas of responsibilities are distributed as illustrated in Figure 2.5. All five RIRs are merged together in what is called the Number Resource Organization (NRO), an unincorporated organization that takes care of joint activities such as joint technical projects and policy co-ordination.

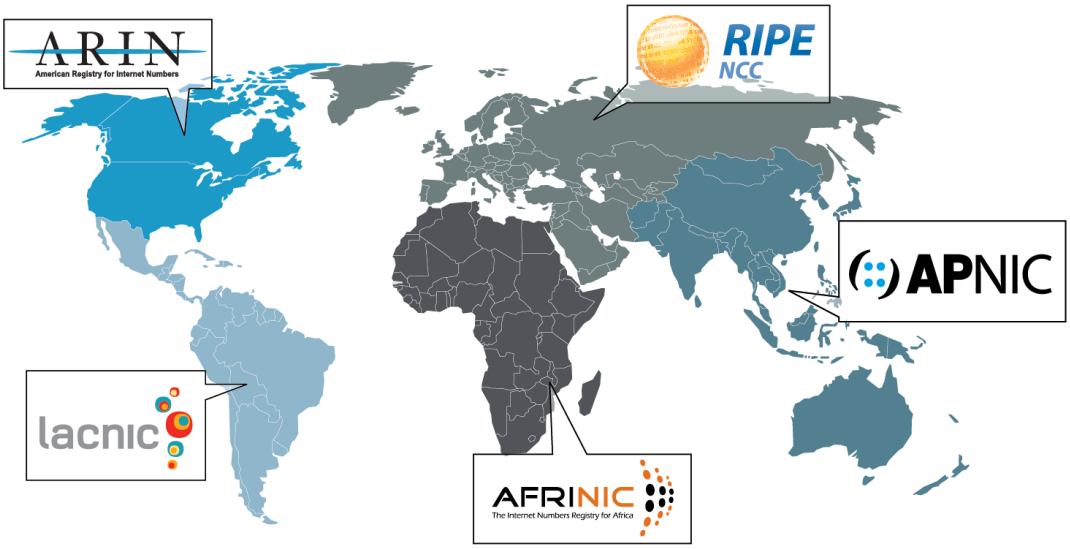

Figure 2.5. Regional Internet Registries map [33].

The knowledge of RIRs covering a physical region and the predetermined address ranges that have been assigned to each of them gives the opportunity to use them as broad landmarks. This concept is used for a monitor assignment technique discussed in Section 4.3.1.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Topology Mapping

Internet topology mapping is an active field of study which helps researchers to understand the network's internal structure, backbone dynamics, management and security risks. Because of the sheer size, complexity, and distributed nature of the Internet, combined with a lack of ground-truth, a complete map of the Internet remains elusive. However, by employing a variety of new techniques, researchers are making continuing progress toward better and more complete snapshots of the Internet topology [34]. But even when most of these techniques consider probing reduction to decrease the load on the network as well as augment mapping speed, the time needed to gather full Internet-size topology maps is still too long to catch small transient dynamics that might reveal properties of interest.

In this chapter, we discuss the work that has been done in the field of Internet topology mapping and active probing. We begin by mentioning measurement platforms available to perform mapping and then analyze the state-of-the-art research in topology discovery. Finally, we end the chapter by explaining SCP, the probing primitive that motivates this thesis.

3.1 Production Topology Measurement Systems

Several research groups have implemented their own independent probing platforms. At an interface-level topology, these production measurement tools rely on the use of active probing in order to gather data. In particular, they utilize the traceroute tool or one of its derivatives such as paris-traceroute. An important distinguishing characteristic between these systems is the network location from which active probes are issued, as the inferred topologies are highly dependent on vantage point. For example, probing from a stable set of fixed vantage points requires deployment of dedicated measurement infrastructures which is costly and may impose a heavy burden on the network. In this section, several current topology projects are reviewed.

Archipelago (Ark) [2] is CAIDA's active measurement platform which replaced Skitter [35]. Ark employs over 65 monitors distributed around the globe divided into three groups to perform team probing, where each team probes independently. As a probing tool it uses Scamper [36], an open-source packet prober designed to support large-scale Internet measurements, which includes implementations of traceroute, ping and alias resolution techniques. The traceroute included in scamper is feature rich, supporting paris-traceroute among its main features. Ark

supports both IPv4 and IPv6. Ark's topology discovery strategy consists in dividing prefixes advertised in the global BGP tables into /24s from which only one randomly selected address is probed. Ark dynamically separates the measuring workload among team members. This parallelization helps to decrease the time taken to probe the /24 address space to about three days.

DIMES [5] is a distributed measurement infrastructure that operates with the help of thousands of lightweight measurement agents deployed around the world. These agents are installed in volunteers' systems and contribute to the research. DIMES data collection methodology relies on traceroute and ping.

iPlane [4] is a scalable service that attempts to predict Internet path performance for overlay services. This topology measuring tool has been applied over services such as content distribution, peer-to-peer filesharing and Voice over Internet Protocol (VoIP), where its predictions have lead to improved overlay performance. iPlane is implemented over the Planetlab [37] infrastructure, and measurements are performed on a daily basis using traceroute. It operates by extracting BGP snapshots from Routeviews servers which cluster paths into groups of prefixes with similar routes. It then builds a structured topology, finding interfaces in the same AS that are geographically collocated. With a compact topology due to the clusters, iPlane can determine diverse link attributes used for overlay performance prediction.

Rocketfuel [6] focuses on measuring router-level ISP topologies. As an infrastructure, Rocketfuel uses the public available traceroute servers at `http://www.traceroute.org` distributed around the world. Rocketfuel extracts Routeviews' BGP routing tables to identify AS-paths that will transit through the target ISP network. This process is called directed probing and also helps to skip unnecessary traceroute probes. With a method called path reduction, Rocketfuel identifies probes that are likely to have identical paths inside the ISP. Then, it uses an alias resolution technique to determine interfaces that refer to the same router. Finally, it determines which ISP the routers belong to by relying on DNS. Although, Rocketfuel has been influential in current topology tools, it is no longer active.

3.2 Current Internet Mapping Efforts

Although in the previous section, we discussed several topology tools; these perform poorly. According to the results obtained by the authors of [38], iPlane and DIMES were able to discover 73% and 11%, respectively, from Ark. Even when Ark appears to have the best perfor-

mance of the three evaluated tools, when compared to an ISP ground truth, Ark fails to capture 27% of the routers. In this section, we analyze recent research efforts on Internet topology mapping that seek to mitigate above mentioned shortcomings.

DoubleTree [8] is a topology probing algorithm that was initially implemented over Skitter, Ark's predecessor. For destination selection it uses the same methodology as Ark, which randomly picks one address per /24 prefix. Doubletree's probing strategy benefits from the tree-like structures of routes on the Internet such as when a single monitor queries a group of destinations. The concept behind this algorithm is to save probes by tracking its progress through the tree, from the leaves towards the root, so that it can keep probing until it encounters a node known to the tree. This strategy uses both monitor and destination as the roots, and its stopping rule is set according to which direction the probing is being performed.

In [9] the authors introduce Cheleby, their active topology measurement tool. Cheleby is an integrated mapping system that performs topology sampling, construction and analysis. It is implemented over the Planetlab network and uses paris-traceroute as its probing mechanism. The process starts by extracting active BGP announcements from `http://www.cidr-report.org`. For each /24 prefix within an active AS, the first allocated IP address is selected as a destination. All targets that belong to the same AS form one destination block. Planetlab nodes are grouped into seven teams based on their geographic locations. Only one monitor per team probes a specific destination block. Each destination block gets probed by each team; therefore, only seven monitors probe each destination block, one at a time. Cheleby has a querying strategy used to decrease the probing overhead. The method starts by sending probes to some destination IP addresses until a complete path from source to destination returns. Then Cheleby modifies the TTL value to start probing from the ingress router, which is the last hop that does not belong to the AS. The other overhead reduction technique is to select one monitor per team to query a particular AS. As mentioned earlier, teams are formed according to geographical location of each monitor; thus, the method assumes that traces follow a similar path towards the destination AS.

One of the first published attempts to map China's Internet topology was presented in [39]. The authors used as a destination selection methodology the extraction of public BGP announcements from servers such as Routeviews and RIPE NCC RIS. To the obtained prefixes, they applied what they called the *nested IP block partitioning*. This finds BGP prefixes residing in another prefix from a different entry and subdivides the larger address space into the min-

imal number of prefixes while preserving the nested structure. The intuition behind this idea is that the small nested prefixes suggest different subnets. For each possible subnet, the first allocated address is selected as the destination for that particular prefix. The probing reduction is performed by having two different sets of nodes. The first one, called the *reach_set*, holds all the addresses a VP has observed during its previous probes, while the second one, called the *source_set*, keeps all the addresses seen in traces sent towards a particular destination block by any VP. Before a monitor probes a block, it verifies whether its *reach_set* and the prefix's *source_set* have an overlap. If the two sets coincide in some elements, it means that a path can be drawn between source and destination and, therefore, that prefix would not be monitored by that VP.

Delay-Guiding Source Selection Method (DGSSM) [10] aimed exclusively at monitor assignment, and its particular contribution is to use as a metric the underlying link delay between the source and a small number of landmarks. It starts by selecting a number of landmarks from a predetermined set defined according to researches such as in [40]. For each monitor, DGSSM calculates the average delay towards every selected landmark. Then, all the monitors are sorted starting with the largest average delay.

In [41] the authors introduce a new metric which they call the Logic Distance (SDIS). Their objective is to lessen the probing load by selecting N sources out of the M available that produce the most efficient output in terms of data learned. SDIS between two sources and a particular destination is calculated dividing the number of hops not shared by paths from either sources towards the destination by the sum of each path's number of hops. The SDIS that is really used is the sum of all individual SDISs for a predefined set of destinations. The basic intuition behind this theory is that if this number is large, the probability to detect more nodes and links is high. Then, by combining SDISs for a set of sources as shown in [41], it is possible to obtain what the authors call the dissimilarity function. As it is explained in [41], the probability of detecting more nodes and links increases when the dissimilarity between the sources grows.

3.3 Subnet Centric Probing

Each of the strategies reviewed in the previous section introduce different point of views on how to perform an efficient Internet map. All of them present innovative ideas of how to avoid bringing a heavy burden generated by active probing into the network, but this still is not a solved issue. For instance, Ark sends a probe to each /24 prefix, which gives in total about 9.5 million addresses to query to cover the whole Internet. This is intuitively a very large spectrum

number and, on average, it takes about three days.

It is shown in [11] that it is possible to generate an initially small set of target addresses which can be dynamically incremented according to the feedback received by sent probes, reducing the time to cover the whole Internet without damaging severely the quality of gathered information. The technique deployed is called SCP, and its fundamental idea is to focus in extracting the internal structure of ASes. From results obtained in [11], it is possible to recover 90% of vertices and edges with only 60% of the load; therefore, by adapting the number of probes to the degree of subnetting, it is possible to avoid wasting them.

3.3.1 Least Common Prefix

SCP as the proposed probing strategy starts by introducing a new concept, the Least Common Prefix (LCP). LCP is the main mechanism by which SCP selects its destinations within the advertised BGP prefix. It works by taking benefit of knowledge of how networks are provisioned and subnetted. As shown in Figure 3.1, given an advertised prefix 16.0.0.0/8, it is easier to believe that A and B belong to different sub-networks than A' and B'.

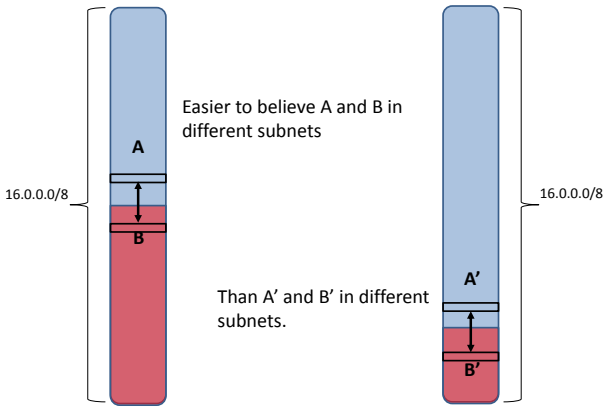

Figure 3.1. Subnetting in advertised prefixes.

LCP divides the advertised prefix, also called *parent prefix*, into two segments. Taking the abstracted network addresses of Figure 3.1 and applying LCP, LCP produces what is illustrated in Figure 3.2 where the parent prefix is separated into two subnets: 16.0.0.0/9 and 16.128.0.0/9. Next, LCP proceeds to select the IP address in the middle of the prefix range as the target address for each subnet.

The partitioning of the parent prefix is always done in powers of two. For example, the corre-

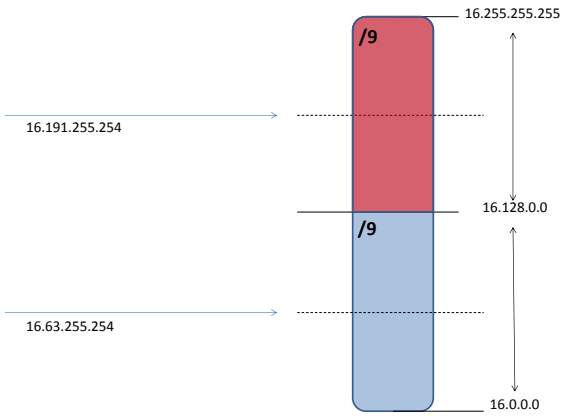

Figure 3.2. LCP divides the parent prefix into two subnets.

sponding output, obtained by continually dividing the original prefix, would be as depicted in Figure 3.3. The destination addresses are once again the center IP addresses.

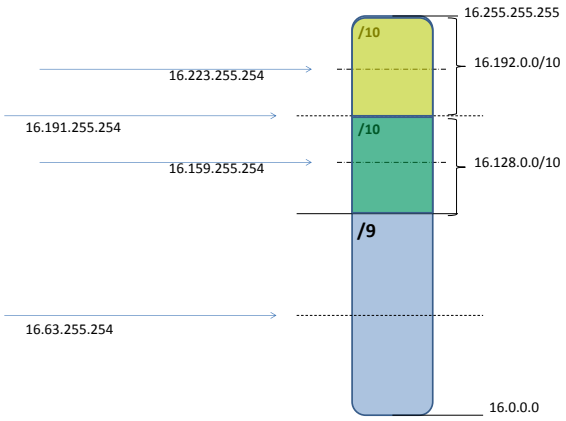

Figure 3.3. Implementing LCP deeper into the parent prefix.

3.3.2 Levenshtein Distance

The Levenshtein Distance (LD) is used in [11] as the stopping criterion to terminate probing a certain branch of a parent prefix. In other words, the LD is the metric that, when compared to a previously determined threshold, tells the SCP that there is not enough benefit in further probing that subnet. What this particular metric does is to compare two traces by counting the number of insertions, deletions and substitutions performed in one trace to change it into the other trace. For example, given the two traces shown in Figure 3.4, two edit operations are needed to change trace two into trace 1: 188.1.50.9 needs to be added and 80.291.253.186 needs to be substituted

by 80.239.128.181. Now, if the stop criterion specifies that the result of the LD needs to be greater or equal to two, the probing algorithm stops probing for that particular subnet given these two traces. Also note that for this metric to be properly implemented, it is necessary that both traces have the same source.

Trace1: 41.208.109.254 139.18.1.254 139.18.122.80 188.1.50.9 80.239.128.181 213.155.135.84

Trace2: 41.208.109.254 139.18.1.254 139.18.122.80 80.91.253.186 213.155.135.84

Figure 3.4. Two traces starting from the same source towards the same subnet.

3.3.3 Random Vantage Point Selection

To assign monitors to destinations the simplest approach is to make random assignments using a uniform distribution. The problem with this is that not all monitors are necessarily used before a repetition in assignment occurs. If we consider that the path followed by two probes from the same monitor towards the same prefix might be similar, then already known nodes and links are revisited, and the opportunity to gain new knowledge is lost. The probability that k /24's are probed by a unique monitor is given by

$$P = \prod_{i=0}^{k-1} \frac{N-i}{N} \tag{3.1}$$

where P is the probability, N is the number of VPs, and k is the number of destinations. Given a /21 prefix and 23 monitors, the probability P that the 8 /24's are assigned a unique VP is 0.25.

3.3.4 Vantage Point Spreading

In contrast to random vantage point selection, [11] proposes and evaluates VPS. The primary benefit of VPS is that it improves the already good results of SCP without extra processing effort. The intuition behind VPS is to use all the available monitors before repeating any so that the paths followed by probes towards certain prefix are as distinct as possible. What VPS does is to assign new destinations to VPs not yet used for probing each parent prefix. In case there are more destinations than monitors for a particular prefix, once VPs are exhausted, the following assignments are performed randomly with a uniform distribution. Basically, the latter can be characterized as a list formed by all the available monitors in random order, where each time a new destination is received, monitors are assigned in a round robin fashion.

3.3.5 Implementation Issues

In [11] it was successfully demonstrated that when using as ground truth data gathered by Ark and with the same data running the SCP algorithm with LD as the stopping criterion, it is possible to obtain 90% percent of vertices and edges while using only 60% of the load. The problem with LD as the metric is seen when implementing SCP in real-time over the Internet due to the existence of load balancing routers. Even when using the paris-traceroute algorithm as explained in Section 2.2.3, the traces are distorted because while using LCP every new route has a different destination address and, thus, the tuples differ one from the other. The result is that the SCP stopping criterion is never reached provided that the load balancing routers change every single time the paths for trace pairs. An example of how pairwise probing works is shown in Figure 3.5. Considering a parent prefix of 16.0.0.0/8 as in the previous examples, the destinations are determined according to LCP and both of them are assigned to the same monitor. When encountering a load-balancing router, their paths diverge and increase the LD. In this example, the LD is equal to four, but since the SCP threshold has been set to two, the algorithm divides the prefix and continues probing.

The destination addresses are selected from the sub-network 16.0.0.0/9 according to LCP. In Figure 3.6, we illustrate the second group of probes sent towards the destination. We use the same monitor for the two groups of probes to avoid more complex diagrams. In a real implementation, the monitor would change for each pairwise probing according to the random strategy or VPS as explained in Section 3.3.4. This model stays true due to the large amount of load-balancing routers on the Internet [30]. As shown in Figure 3.6, once again the LD is distorted by load-balancing routers; thus, it continues probing even when the information gain is not significant. This distortion forces the algorithm into an endless probing loop.

Finally, [11] evaluates SCP and VPS in isolation without considering their interaction. Without modification, SCP and VPS cannot be combined due to SCPs reliance on pair-wise path comparison between traces that originate from a common VP.

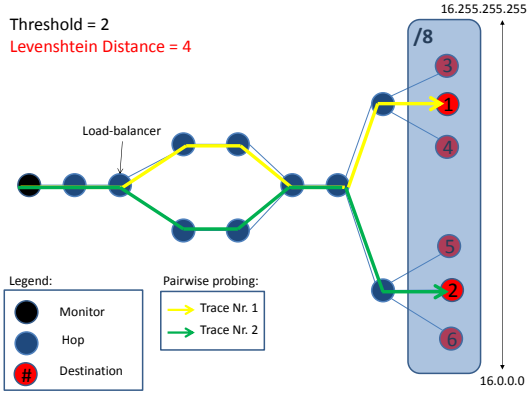

Figure 3.5. Pairwise probing towards parent prefix 16.0.0.0/8.

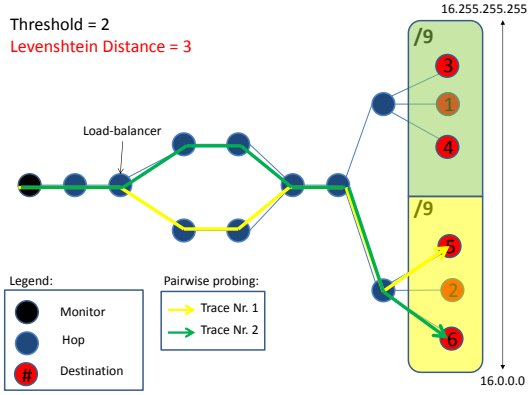

Figure 3.6. Pairwise probing towards subnetwork 16.0.0.0/9.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Theory and Design

In the preliminary work of [11], it was shown that in terms of probing budget there is benefit in intelligently selecting the set of vantage points using a strategy such as VPS and determining the destinations with an algorithm like LCP. However, the practical existence of load-balancing routers on the Internet prevents the algorithm from being implemented directly. The primary limitation of LCP is its use of LD as a metric since it gets distorted. This distortion can lead to an infinite probing loop, which performs even worse than randomly choosing the set of destinations.

The objective of this chapter is to address issues introduced by load-balancers by developing alternative solutions that permit an online implementation of SCP. The main strategy that is introduced is called New Interface Discovery (NID), which measures the information made available by each new probe. In addition, two new methodologies are studied that try to solve VPS deficiencies.

4.1 Restructuring SCP

In this section, we start by identifying a three-step process which helps us think of SCP as a block structure with different layers. A layered structure allows different methods without affecting contiguous layers, thus, making it easier to analyze each step of the process by itself permitting different combinations of diverse strategies. As shown in Figure 4.1, the three blocks structure used in this thesis' SCP implementation are destination selection, monitor assignment and stopping criterion.

Destination selection so far has been successfully implemented through the LCP algorithm and remains the same during the course of the present work. Although the strategy presented in Section 3.2 called nested IP block partitioning is fully integrable with LCP and might present some benefits to SCP, the analysis and evaluation of integration between the two is left for future work.

Since LD presents issues mentioned in Section 3.3.5, such as dependence on pairwise probing and distortions by load balancing, which currently do not allow SCP an online implementation, we begin by analyzing possible solutions to this problem.

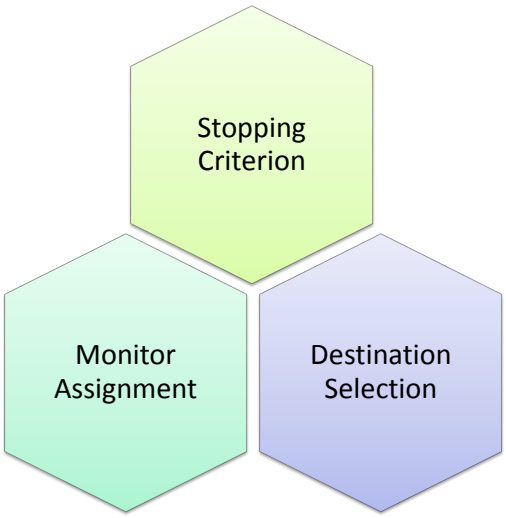

Figure 4.1. SCP block structure.

4.2 Stopping Criterion

The LD has several deficiencies in addition to the non-stopping issue outlined in Section 3.3.5 that hinder SCP's full potential. For instance, pairwise probing is agnostic of previous probes by throwing away information gathered earlier and not counting it as part of the metric. Pairwise-probing also forces the use of the same router, which is necessary to properly calculate LD, wasting the opportunity of using different monitors per probe and maximizing the benefit obtained from each trace.

4.2.1 New Interface Discovery (NID)

We present NID as an alternative solution to LD, conceptually it is very simple but very powerful in relation to what it allows SCP to do. This new metric focuses on discovering the internal structure of the destination prefix. To do that, the method remembers all the interfaces seen inside the destination AS by each probe sent toward the parent prefix. For each new trace, only the trace's hops inside the destination AS are counted and compared with the remembered set of interfaces. To determine which hop is inside the destination AS, each hop's ASN is contrasted with the corresponding destinations address ASN. The ASNs are obtained from the Routeviews servers. The number of new hops found in a trace is compared with a predefined threshold

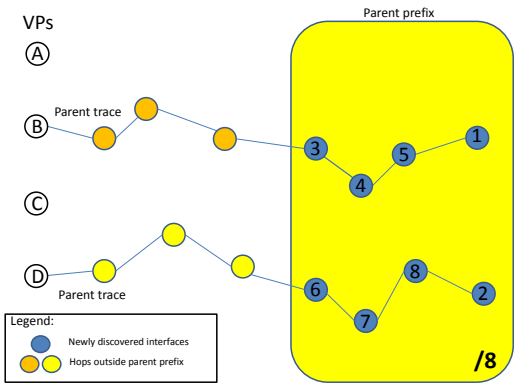

Figure 4.2. NID parent traces.

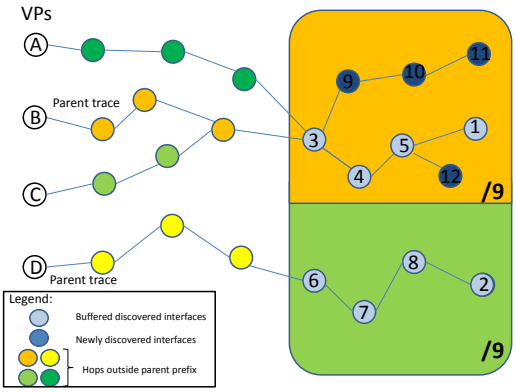

Figure 4.3. NID sub-network probing.

to determine whether the algorithm should continue dividing the prefix into subnetworks. The traces that describe paths followed by probes sent towards the parent prefix are called parent traces.

To provide intuition and explain NID, in Figure 4.2 we have illustrated two parent traces sent towards the parent prefix represented by the shaded yellow area whose mask is a /8. In the figure, the circles with letters in them represent available monitors for use as sources. Numbered blue circles indicate hops inside the destination AS. Circles 1 and 2 represent the selected destinations by LCP. As mentioned in Section 3.3.1, LCP selects the address in the middle of the address space of the smaller subnet, in this case the /9 prefix. Orange and yellow circles indicate hops outside the destination AS.

The first benefit from using NID is that the number of probes sent at any particular time to a prefix does not need to be two, as we no longer make use of LD. Instead, NID can send, and compare, any number of probes. Although the figure only shows two traces for clarity, varying the number of probes sent per prefix is possible because the metric compares the number of newly discovered interfaces with the buffered total and not to a particular trace. In the example, four new interfaces have been discovered by each probe, and if a threshold of two is considered, the SCP algorithm divides the parent prefix into two /9 sub-networks and continues probing them independently.

Continuing with the same example, in Figure 4.3 we depict how two probes are sent towards destinations in the /9 sub-network as selected according to LCP. The dark green trace has

discovered three new hops. This is greater than the threshold; therefore, SCP continues probing that prefix. On the other hand, the light green trace found only one new hop, which is below the threshold. Thus, that prefix is not probed any further.

4.2.2 New Capabilities

The most defining characteristic of NID is that it focuses on the AS's internal structures, which basically goes hand in hand with SCP's primary intention. While NID is concerned with counting the number of new interfaces discovered, it overcomes any problem generated by load-balancing and actually ends up benefiting NID by revealing previously unseen vertices and edges. NID allows source distribution by focusing on a target AS as opposed to pairwise probing. Recall that NID permits the comparison of traces originating from different VPs. Employing multiple VPs as part of SCP naturally is valuable because doing so helps discover AS ingress points. In contrast, when all probes to a destination originate from the same VP, all probes enter the AS via a single ingress. As was mentioned earlier, the number of probes no longer needs to be two since it admits sending as many probes as needed. That number depends on the probing strategy used and the results of the traces. NID maintains state by taking into account previously sent probes. Finally, NID leaves room for future improvements such as including edges as part of the metric.

4.3 Monitor Assignment Techniques

As explained in Section 3.3.4, VPS is a strategy for monitor assignment for SCP. VPS can be thought of as providing a pool of potential monitors to SCP now that NID has decoupled the selection of VP from its stopping criterion. Without an extra cost VPS, can outperform the more simplistic random assignment technique. Nevertheless, VPS still has some deficiencies that leave space for improvement. This monitor spreading strategy commits every single available monitor before repeating one into the same prefix but it does not specify the order in which those monitors have to be used. Intuitively, by starting with specific monitors towards a determined network, SCP will be more successful in discovering ingress routers. Therefore, probes will be able to find more new interfaces inside the destination AS and also avoid an early termination of that particular probing branch. Additionally, some monitors will be relatively farther in terms of hops from the destination AS, and the number of vertices and edges learned will be greater.

In order to determine whether the first few probes sent into a parent prefix are important, we conducted a simple experiment. The objective of the experiment was to find the number of probes sent per prefix out of a set of 1500 ASes extracted from Routeviews. The total number

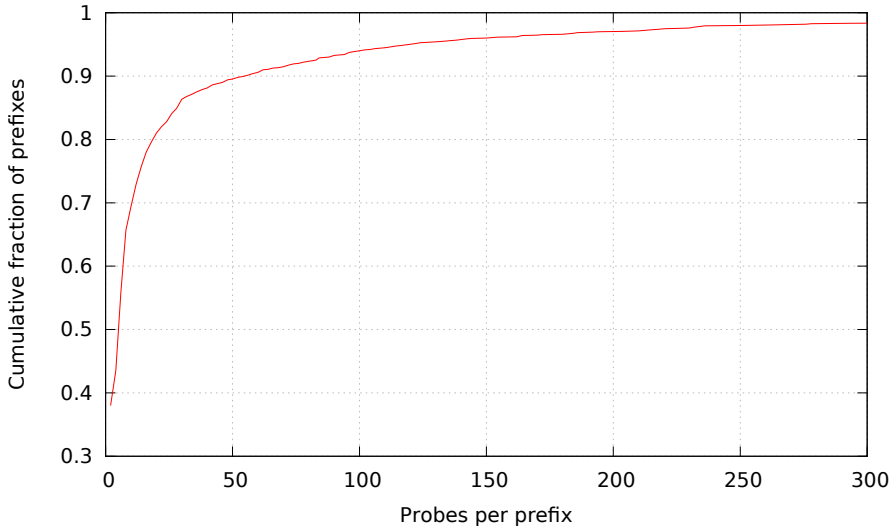

Figure 4.4. Cumulative distribution function of the number of probes sent per prefix.

of monitors available for use was 60, and the assignment technique employed was the random method as explained in Section 3.3.3. In Figure 4.4 the resulting cumulative distribution function is shown, where about 90% percent of the prefixes send only 50 probes before SCP finishes probing it. The figure has been zoomed-in to make it easier to focus on the important part. Considering that 60 monitors were used, more than 90% of the prefixes did not get to use all the available monitors even if no monitor was assigned twice from a single AS. This result shows that it is important to choose the order of selection of the monitors wisely, otherwise it might mean an early termination for probing that particular prefix when there is still information to extract.

4.3.1 Maximum Coverage method (MAX)

The MAX method is a rank ordering strategy whose aim is to carefully decide the precedence in which monitors are selected by enforcing the maximum predicted path diversity. As input, MAX takes previously collected probes, e.g., from a prior round of probing. This technique's purpose is to enhance the amount of new information learned by each individual trace. It can be seen in Figure 4.5 (a), when choosing monitors 1 and 2, both probes in this example follow a similar path with shared hops toward the same prefix. Also, the monitors are located too close in term of hops, potentially wasting the opportunity to find new vertices and edges before reaching the destination AS. On the other hand, as shown in Figure 4.5 (b), both paths are maximized by hop distance, from monitor to prefix and from monitor to monitor, thereby discovering more vertices and edges. Finding different ingress points into the destination AS is a side beneficial

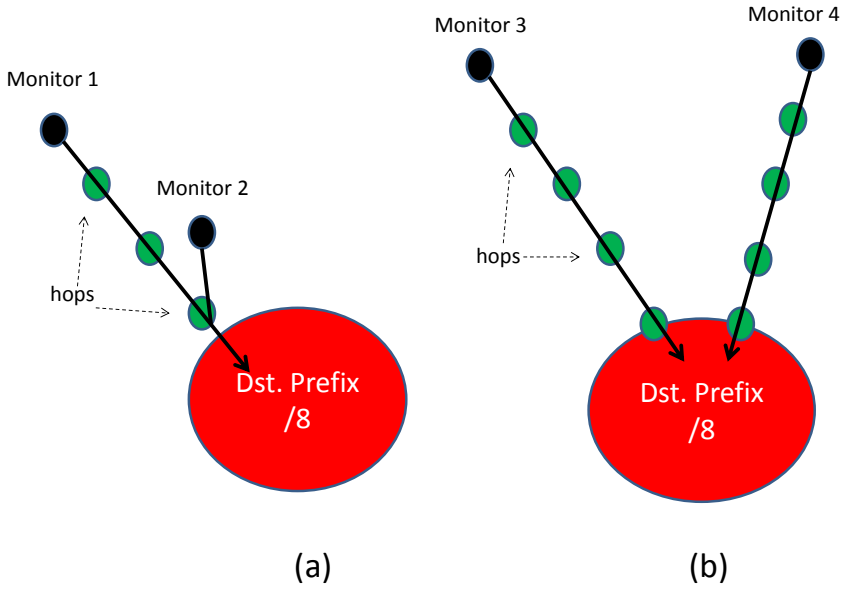

Figure 4.5. Intuition behind the maximum coverage monitor assignment technique.

effect in MAX.

Rank ordered lists are formed by MAX using input trace data that might have been gathered by doing a pre-probing round on the Internet or by using previously probed data. Therefore, a database of traces is needed before implementing this kind of process, which is performed offline as a precomputation step. The MAX algorithm basically forms lists which SCP can potentially exploit.

Given that the intuition behind MAX is to order monitors according to their distance in IP hops from the destination prefix as well as from each other, it is necessary to establish a criteria to determine the appropriate relative values. For MAX, we explore whether logical landmarks can help to successfully classify monitors according to their hop distance to specific prefixes. As explained in Section 2.3, RIRs allocate IP addresses in specific regions around the world. In particular, we know that RIRs get addresses assigned in blocks of /8, making that prefix range a natural logical landmark selection to explore. Since addresses in the /8 prefix are likely to be located in the same physical region, it is possible to calculate the hop distance from every available monitor with the same IP address selected from that prefix range and then proceed to order them according to the MAX algorithm. This is explained next.

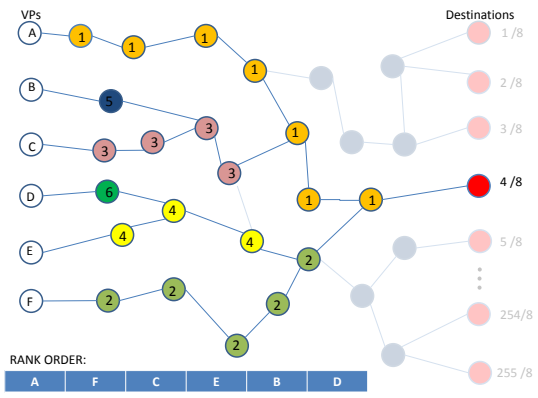

Figure 4.6. Maximum coverage algorithm.

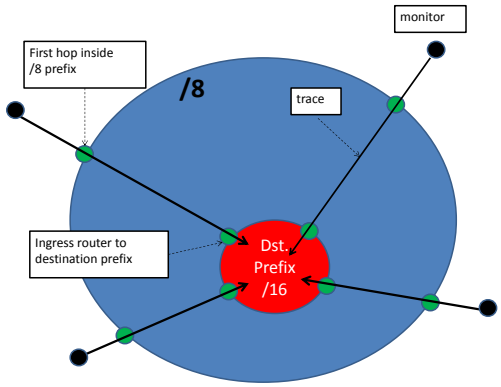

Figure 4.7. Intuition behind the IPS algorithm.

By probing one single address per /8 prefix with every available monitor, it is possible to build a rough approximation of the macro Internet map. That approximation allows us to form a rank order list for each /8, starting by finding the farthest monitor. All the hops that belong to the farthest node's path are marked down as used by that first monitor as seen in Figure 4.6 and are not counted again. The second monitor in the list is selected from available VPs whose shortest path to the destination has the most unseen hops. The next monitor on the list is selected following the same procedure. In the example shown in Figure 4.6, monitor A is the farthest in terms of hops from the destination, then follows monitor F, C, and so on. The hop distance comes from the shortest path as calculated by the Dijkstra algorithm. In the case of two paths with an equal number of hops, a simple heuristic is used. This is an iterative process individually performed for each /8 prefix.

4.3.2 Ingress Point Spreading (IPS)

MAX's objective is oriented to finding as many nodes and links outside the destination AS as possible; thus, obtain a richer global topology map. Finding ingress points to destination prefixes is a side effect. On the other hand, IPS solves the problem the other way around. This monitor assignment strategy aims at finding ingress routers to target networks, which in turn leads to path diversity and variety of vertices and edges. IPS is the natural complement to SCP and NID as it is focused in finding specific characteristics about target networks' structures. The basic intuition behind IPS is to increase the probabilities of finding ingress routers to a network by using paths already known to probes gathered in past cycles. Specifically, we use IPS to

explicitly attempt to utilize all known ingresses points into the logical landmark.

An easy way to order monitors according to their chances of using a specific ingress router is by analyzing pre-probed databases from where to find paths from available monitors to destination prefixes. Then, it is possible to form a group of monitors that are found to use a specific ingress router to the destination network. All the monitors in the group should not have, in any of its analyzed traces, any of the ingress routers considered by other monitors in the group, trying to secure that specific path. In case a second monitor is found to access the network using an already considered ingress router, this monitor is left out of the main group since the algorithm forces it to use all the known ingress routers first before repeating. The monitors part of the group are assigned first in order to have higher chances of entering the prefix through different gateways.

Ideally, IPS determines, for each BGP prefix, a rank-ordered set of VPs to ensure that all known ingresses into the prefix are utilized. However, in this thesis, we approximate this full IPS functionality by finding ingress points into the logical /8 landmarks. IPS, instead of keeping track of ingresses to all possible subnetworks, searches in the pre-probed data for gateways to the /8 prefix of the destinations address. As shown in Figure 4.7, using this approach it is possible for us to know what path has a greater chance to be followed by a probe sent from a specific monitor to a particular AS. Focusing on the ingresses to the /8 prefix decreases the algorithm's overhead. The process of forming the group is followed the same way as described in the ideal case.

Since IPS is a rank ordered list, it is built from a pre-probed dataset. From the pre-probing round, from each trace a tuple of three elements is extracted: source, ingress router, destination. By ingress router we mean the last hop before starting with nodes inside the /8 prefix to which the destination belongs. Then, the algorithm simply scans over the tuples, finding which destinations belong to the selected destination prefixes and associating a monitor, an ingress router and a target prefix.

In Figure 4.8 can be seen the results of analyzing traces from an example pre-probing round that utilized six VPs to trace to seven destinations. The numbered dots represent VPs, the blue dots refer to all the various hops and the red dots the destinations addresses. The light purple area traces out a /16 prefix, which in this example is the destination prefix, and the blue area marks out its corresponding /8 prefix. A /8 prefix does not necessarily have all the interfaces attached together. For the given data, only three out of the six available monitors have paths directed

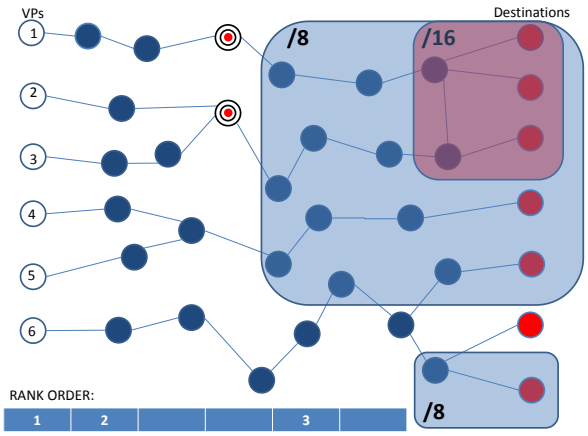

Figure 4.8. IPS building a rank ordered list (initial step).

toward the prefix under scrutiny. VP1 has its own ingress router into the /8 prefix, indicated by a bullseye dot, but VPs 2 and 3 share the same ingress router. Since we are looking for unique ingress routers, a simple heuristic decides which one would have a higher priority in the list. In the example, VPs 1 and 2 go at the beginning of the rank order, leaving VP3 to the end.

Note that, in this example, only three of the VPs in our known set of pre-probed paths elicit responses from router interfaces within the target destination prefix. However, the SCP algorithm may require more than three traces to cover the prefix. Therefore, we wish to ensure that the IPS algorithm provides a rank-order over the complete set of possible VPs. Next, we show how to expand the scope of the destination prefix in question to iteratively create a "virtual prefix," where ingresses into this virtual prefix drive the remainder of how IPS populates the rank-ordered list of VPs.

As was explained in Section 3.3.4, the hypothesis behind VPS is that the more variety of VPs used, the more efficiency of sent probes and quality of gathered information. In that sense, a potentially useful way to include those monitors which the pre-probed data does not consider with paths into the target prefix is to increase the destination network's address space by decreasing the prefix's mask. As illustrated in Figure 4.9, the prefix is decreased by one, from /16 to /15, and if monitors that are not part of a tuple yet are found to have a path toward the /15 prefix, their tuple gets registered. The monitors get added to the rank order list following the same criteria as before, sending the repeated monitors to the back of the list. The process continues with the /14 prefix and so on, until all the monitors form part of the list.

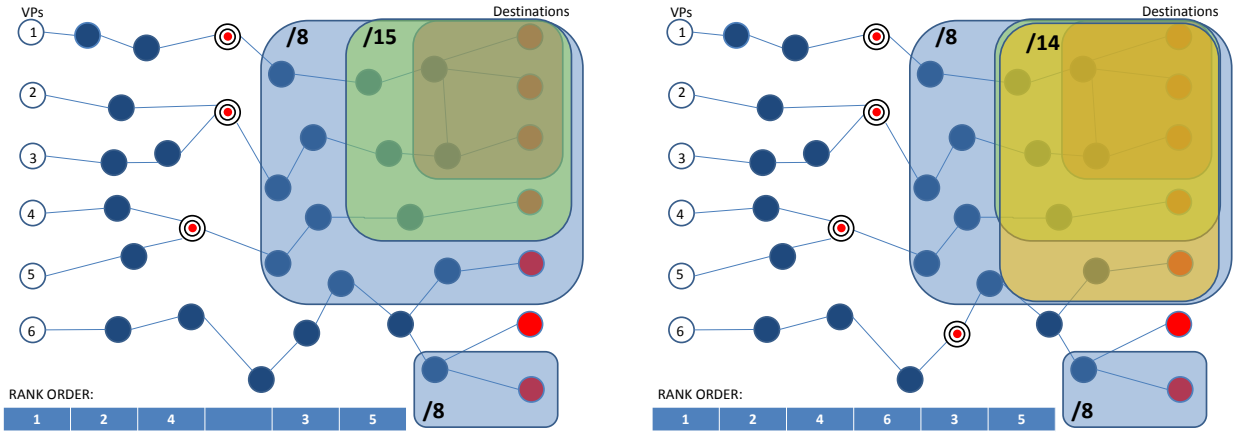

Figure 4.9. IPS expanding coverage for full VPs usage.

CHAPTER 5:
Implementation and Results

This thesis' core ideas, which are conducive to addressing the objectives presented in Section 1.3, were presented in Chapter 4. In this chapter, we discuss the specific methodology and implementations employed as well as comparative performance results observed among the different strategies we implement. We mention the design considerations taken during the implementation phase and also describe the algorithms themselves. The chapter ends with an analysis of the results obtained when implementing the proposed probing strategies.

5.1 Design Considerations

There are several design considerations that must be taken into account to deal with all the difficulties presented during algorithm run-time. The main consideration, previously described in Section 4.1, is the flexibility needed to evaluate different concepts and strategies when determining monitor assignment and destination selection.

Given the various topology tools for active and passive measurements available to accomplish the execution of SCP over the Internet, we had to make several decisions on how we chose to implement and test our proposed new strategies. For instance, it was necessary for us to define the servers from where to obtain prefixes announced by ASes and what network analysis project to use as an active probing platform.

5.1.1 AS Observation

AS-level data is the first element needed by SCP, since this data allows us to properly select destination addresses that target particular network aggregates. SCP is able to adapt to subnets' different sizes by extracting their internal structure and driving future probes depending on feedback received from previous results; therefore, SCP does not have strong reliance on the BGP vantage point.

As mentioned in Section 2.2, there exist diverse projects from which to obtain BGP prefix announcements. Data from each project varies depending on its point of view and how paths are aggregated. In terms of what is needed to feed SCP with proper information, all these strategies are adequate. Therefore, for practical reasons, BGP prefixes are obtained from the

Routeviews servers, leaving the comparison of results of LCP using other prefix sources for future work.

5.1.2 Probing Platform Selection

The selection of Ark as our active probing platform over those other methods explained in Section 3.1, such as DIMES and iPlane, is based in two factors. First, results presented in [38] favor Ark over other methods. Second, Ark has a very convenient Application Programming Interface (API) for easy primitive implementation.

Ark's API facilitates easy development and rapid prototyping – important attributes as the characteristics of our primitives evolve. The API allows a high-level of abstraction, which in turn leads to rapid prototyping. This application interface is programmed using Ruby, an open-source programming language with convenient characteristics for API programming. Ark provides tailored libraries for controlling Scamper's tools to perform parallelized traceroute and ping measurements; thus, allowing researchers to evaluate and implement complex probing strategies easily.

The Ark server offers a service called topo-on-demand which lets clients connect to a single point instead of needing to login into each monitor independently. A client can perform multiple parallel queries to monitors through Ark server, which in turn returns measurements results asynchronously. Each Ark monitor is running Scamper and, thus, performs paris-traceroute to every routed prefix. The maximum number of probes an Ark monitor handles concurrently is 75.

5.2 Implementation

From a system's point of view, the heart of the model lies at the SCP server as is shown in Figure 5.1. The three step process starts by obtaining destinations to probe by querying the Routeviews servers for available prefixes to set as targets. With randomly selected destination prefixes, LCP generates addresses to probe within those prefixes following a deterministic process. Next, each destination is assigned to one of the available monitors following the procedure explained in Chapter 4. SCP evaluates probing feedback to determine when the stopping criterion is met.

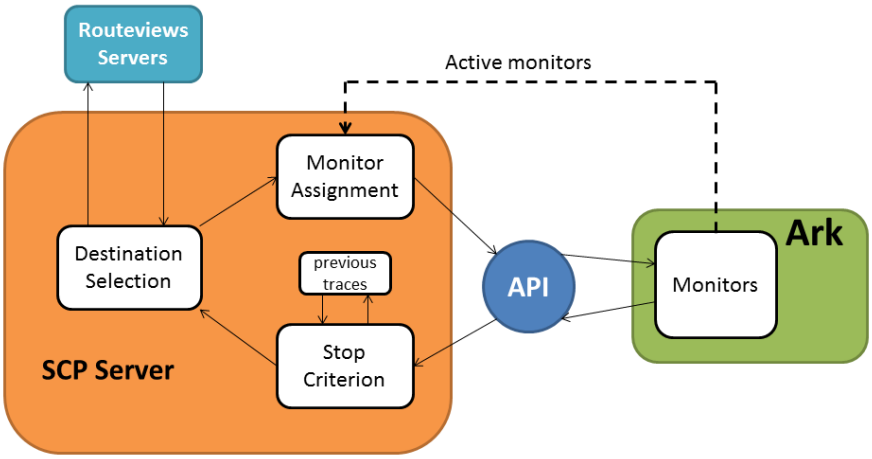

Figure 5.1. SCP system overview.

5.2.1 Destination Selection

In our testing, SCP randomly selects chosen BGP prefixes from the Routeviews servers. We use a random subset to facilitate the initial experiments in this thesis. In future work, SCP may operate over all advertised BGP prefixes. For prefix ranges that are found to reside inside other prefixes, the one with the most path aggregations is chosen. Related works reviewed in Section 3.2 take more in-depth approaches to destination selection than LCP, such as nested IP blocks, from which LCP might benefit in the future.

5.2.2 Probing Strategy

Attributes given by using NID as the stop criterion allow several elements to be tuned which, in turn, permits the evaluation of different probing strategies. For instance, the first attribute easily modified is the stopping threshold. The smallest possible threshold in NID is one. This means that so long as a single new interface is discovered by a trace, the method continues to probe that particular prefix. Parent traces are treated as an exception to this rule. Since traces arrive asynchronously, the first one to arrive is favored as the discoverer of new hops, thus, leaving following traces that return with these already known hops without possibility continuing probing its range of the prefix. So as not to just discard a section of the prefix only because its probe arrived later, exclusively for the parent probes, NID counts all the interfaces

inside the target AS as new.

Destination addresses, when reached, are not counted as part of the NID metric but are saved as known hops for comparison with future hops.

Another characteristic of SCP that may be tuned is the number of probes sent to prefixes. When using LD as stopping criterion, SCP was forced to use two probes per prefix in order to permit pairwise edit distance comparison. In contrast, NID allows number of probes to be sent to prefixes. In some of the experiments that are discussed in Section 5.4, we have sent different numbers of parent traces depending on the implemented strategy. For easy prefix division, the number of probes sent has been conveniently approximated to the following integer power of two. We have provided the pseudo code that implements NID in Table 5.1.

5.2.3 Monitor Assignment

To implement any of the monitor assignment techniques, we have to determine the current set of available monitors. We execute this procedure by sending 10 probes from each listed monitor in the Ark registry to 10 randomly chosen addresses, and those monitors that are returned ten traces are selected as part of the set of available monitors. In this section we go over the specific details of how MAX and IPS perform their rank list ordering.

MAX as an algorithm does not actively probe the Internet. Instead, it runs over pre-probed data in order to build rank ordered lists of monitors with respect the predetermined /8 prefix as logical landmark. If not all monitors have complete traces to the specified address within the landmark, paths are filled by finding the shortest path. The algorithm is presented in Table 5.2.

The IPS monitor assignment strategy similarly uses prior rounds of traces but attempts to utilize diversity as measured by ingress routers to the /8 prefix virtual landmark. Ingress routers in this context correspond to the last hop before starting interfaces within the /8 prefix. It is not necessary for the /8 prefix to correspond to a single network since the intuition behind IPS is that discerning which routes have followed probes in the past might indicate paths that will be followed in the future, maximizing information learning. In Table 5.3 we show the pseudo-code that describes the execution of the IPS algorithm.

5.2.4 Implementation Issues

Some of the issues that we had to deal with when conducting measurements were sent probes that did not return because of measuring tool exceptions, incomplete traces due to NATs and

Table 5.1. Pseudo-code describing NID algorithm.

Input: p/m: destination prefix / mask
M: set of monitors
τ: threshold of new interfaces
Output: T: set of path traces; initially empty

Algorithm 1: $NID(p/m, M, \tau)$

1: $I = empty\ set$ // global variable
2: $T = empty\ set$ // global variable
3: $SCP(p/m, M, \tau)$

Algorithm 2: $SCP(p/m, M, \tau)$

1: $A = ASN(p/m)$
2: $(d_1, d_2) = determine\_targets(p/m)$ // LCP algorithm
3: $(m_1, m_2) = assign\_monitors(M, p/m)$ // Random, VPS, IPS, MAX.
4: $t_1 = trace(m_1, d_1)$
5: $t_2 = trace(m_2, d_2)$
6: $T = T \cup (t_1 \cup t_2)$
7: $t_1' = interface\ i\ in\ t_1 \mid ASN(i) = A$
8: $t_2' = interface\ i\ in\ t_2 \mid ASN(i) = A$
9: **for** $j = 1$ **to** 2 **do**
10: **if** t_j' *is parent_trace* **then**
11: **if** $length(t_j') > 0$ **then**
12: $I = I \cup t_j'$
13: $SCP(p/(m+1), M, \tau)$
14: **else**
15: **if** $|t_j' - I| \geq \tau$ **then**
16: $I = I \cup t_j'$
17: $SCP(p/(m+1), M, \tau)$

firewalls causing probing to end prematurely and unresponsive routers that contribute to topology inference errors.

Handling non-returning probes was initially approached using a method called "hovering." This method sends probes to the nearest adjacent addresses still not selected for probing every time a specified waiting time for probes to return with results expired. This method is implemented

Table 5.2. Pseudo-code describing MAX algorithm.

Input: G: graph built by pre-probed traces
M: set of monitors
P: set of each /8 prefix
Output: R: set of ranked monitor lists per /8 prefix

Algorithm: $MAX(G,M,P)$

1: **for each** $prefix$ **in** P **do**
2: $Y = \emptyset$ // Set of shortest paths from monitors to selected destination
3: $dest_{prefix}$ = **select** one destination in P from G
4: **for each** monitor **in** M **do**
5: $t = shortest\_path(monitor, dest_{prefix}, G)$
6: $Y = Y \cup t$
7: **for each** monitor **in** M **do**
8: $m = max(Y)$
9: $append(m)$ **to** r_{prefix}
10: **delete** all hops in m from all traces in Y
11: $R = R \cup r_{prefix}$

$$d' = i(-1)^i + d \qquad i = 0,1,2... \tag{5.1}$$

where d is the current address targeted by non-returning probe, d' is the nearest unused address, and i is the index of sent probes due to current non-returning probes.

Fortunately, the non-returning probes exception due to an Ark misconfiguration has been overcome by CAIDA administrators, and hovering has not been needed to obtain the results presented in this thesis.

Based on our empirical evidence, probes not completing the path all the way to the destination address amount to 82% of the total number of traces. Incomplete traces can occur if there is NAT or firewall filtering as well as inappropriate destination selection. The former has no solution for current discovery techniques, and the latter is left for future work. Traces that do not provide any information about the target AS might produce an early termination. Trace incompleteness particularly affects NID results when it refers to parent traces.

Table 5.3. Pseudo-code describing IPS algorithm.

Input: T: set of pre-probed traces
M: set of monitors
p/m: set of prefixes / masks
Output: R: set of ranked monitor lists per prefix / mask

Algorithm: $IPS(T, M, p/m)$

1: $I = \emptyset$ // set of tuples formed by {monitor, first hop in destinations /8 prefix, and destination} from each trace
2: $J = \emptyset$ // set of tuples formed by {monitor, first hop in /8 prefix from traces with destination inside p/m and mask
3: **for each** $trace$ **in** T **do**
4: $(src, dst) =$ source and destination of trace
5: $D =$/8 prefix of dst
6: $hop =$ **find** first hop in $trace$ inside D
7: $I = I \cup (src, hop, dst)$
8: **for each** $prefix$ **in** p/m **do**
9: $r_{prefix} = \emptyset$ // array of tuples containing monitor and first hop
10: **for each** $monitor$ **in** M **do**
11: $flag = False$
12: **for** $mask = m$ **to** 8 **in** $steps = -1$ **do**
13: **if** $flag$ **then**
14: **break** // continue with next monitor
15: **for each** tuple **in** I **that contains** $monitor$ **do**
16: $D =$/mask_prefix of dst in tuple
17: $P =$/mask_prefix of p
18: **if** $D == P$ **then**
19: **append** $(src, hop, mask)$ **to** r_{prefix}
20: $flag = True$
21: **delete** tuples from r_{prefix} with repeated monitors and first hops
22: $sort(r_{prefix})$ // by mask from higher to lower
23: $J = J \cup r_{prefix}$

Anonymous or unresponsive routers, usually characterized by the symbol '*', are not considered for any of the evaluations. In this thesis, anonymous routers do not count as vertices, edges or ingress routers. If an unresponsive router is found just before the first target AS hop, then that probe does not add to the count of ingress routers.

5.3 Methodology

To determine whether the proposed strategies are effective, we need to execute appropriate tests and evaluations that can be used later to compare datasets gathered with different techniques. Our first test consists in evaluating the feasibility of effectively implementing NID on the Internet which was left pending in [11]. The test consists of probing 500 randomly selected prefixes between /16 and /20. We have chosen these prefix ranges since we believe that they are large enough for the specific characteristics of our techniques described in Chapter 4 for proper testing and evaluation. The chosen monitor assignment technique for this test is random assignment to assimilate to Ark's probing strategy. The monitors that we are use for this and all subsequent evaluations are shown in Table 5.4. These monitors have been picked because of the reliability they have shown in previous probing cycles. In general, due to the maximum of 75 probes that an Ark monitor can control, we have kept our number of traces in flight to 2700; that is, depending on the monitor assignment method, about 50 probes sent per monitor at the same time.

Table 5.4. List of the 54 Ark monitors used in algorithm evaluations.

bcn-es	pry-za	bjl-gm	mty-mx	sea-us	zrh2-ch
wbu-us	snn-ie	cmn-ma	sin-sg	jfk-us	san-us
ams-nl	vie-at	per-au	sao-br	dub-ie	sao2-br
rno-us	sql-us	fnl-us	yto-ca	fmo-de	rek-is
bma-se	she-cn	mel-au	lej-de	tpe-tw	pna-es
laf-us	sjc2-us	scl-cl	bwi-us	ams2-nl	nap-it
osl-no	hkg-cn	nrt-jp	yyz-ca	cdg-fr	mry-us
cbg-uk	eug-us	iad-us	cjj-kr	zrh-ch	amw-us
cgk-id	pek-cn	syd-au	ord-us	hnl-us	her-gr

The results given after implementing random monitor assignment, with a threshold equal to one

and two parent probes can be seen in Table 5.5 and Figure 5.2. Ark data corresponds to the data gathered by monitors shown in Table 5.4 during one day of probing using Ark's methodology. Only those vertices and edges that belong to traces sent to the same set of prefixes are counted. For both sets of data, Ark's and SCP's non-responsive routers, as explained in Chapter 2.2.3, are not considered as vertices or edges. These two parsing methodologies stay the same for every test performed in this thesis. The NID threshold is set to one, which is the lowest possible threshold. The total number of unique vertices discovered by SCP using NID as its stopping condition represents 59% of Ark's data.

Table 5.5. Results comparison for different probing strategies.

	Ark	Rand	VPS	MAX	IPS	IPS++
Probes sent	105,501	19,746	20,140	19,780	20,452	39,758
Unique vertices	49,331	27,793	28,114	28,040	28,650	32,870
Unique edges	114,025	76,435	77,853	76,957	79,176	118,441
Ingress Routers	8,916	1,765	1,794	1,798	1,870	3,433
Hops within AS	12,835	3,747	3,930	4,050	3,925	7,246

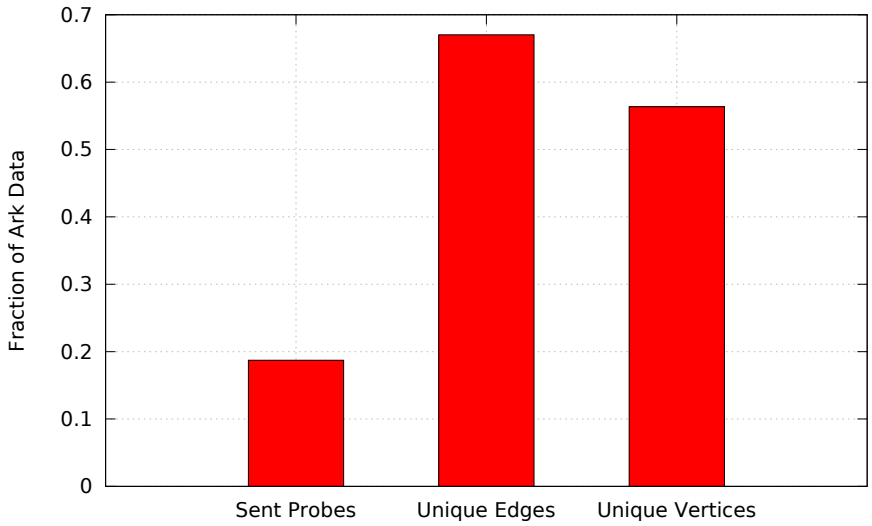

Figure 5.2. SCP compared to Ark data using NID algorithm.

Before deciding the benefit of introducing the overhead that rank ordering provides to our probing strategy, it is necessary to evaluate if there is space for improvement to the VPS monitor assignment method. We consider VPS instead of random assignment, since in [11] VPS was shown to perform better.

To determine whether the opportunity exist to perform better than the results achieved via VPS monitor assignment, we examine the same 500 prefixes evaluated earlier in ten different continuous probing cycles. We seek to understand the influence of VPS's randomness on the number of vertices learned in different cycles. By establishing upper and lower limits on the number of vertices discovered, we approximate the possible gains of more intelligent monitor assignment. Even further, if we could determine theoretical upper and lower bounds, it would be possible for us to compare every monitor ordering algorithm that we develop against a theoretical maximum and to know how well it is performing. However, obtaining these theoretical values is beyond the scope of this thesis.

As shown in Figure 5.3, there is a difference of over 1% between the obtained maximum and minimum number of vertices found across the ten cycles simply due to the randomness inherent in VPS. If we consider that one day worth of Ark data for 50,000 randomly chosen prefixes, that is, approximately 10% of all announced BGP prefixes on the Internet, can have over one million unique vertices according to our analysis, that variance of 1% amounts to over 10,000 interfaces that are not mapped. Thus, we find that intelligent monitor selection is an important task.

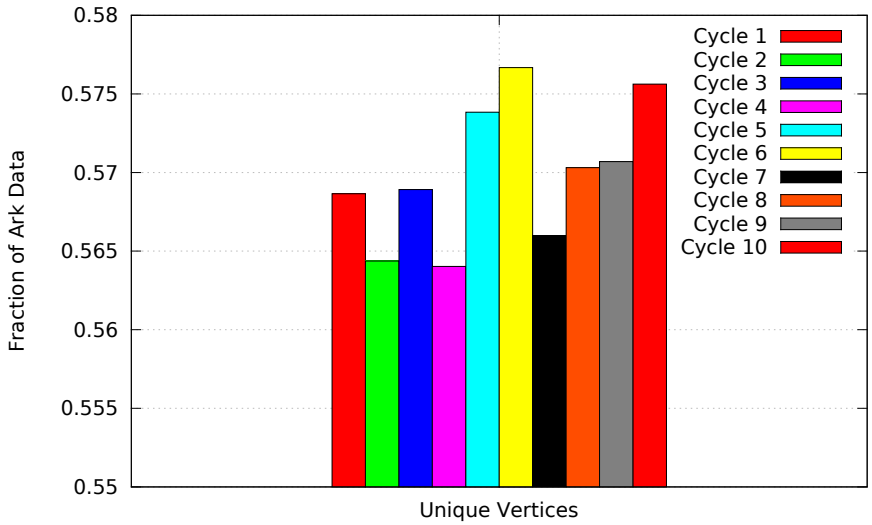

Figure 5.3. 10 cycles of SCP compared to Ark data.

5.4 Results

In this section we proceed to analyze the results obtained across the different monitor assignments methods. To maintain dataset comparison as accurately as possible, we have continuously

produced all probing rounds during the same day, targeting the same 500 destination prefixes previously used and with the NID threshold set to one. The results of the VPS cycles probed earlier have been averaged for comparison purposes.

As shown in Figure 5.4, the random method, VPS, MAX and IPS discover, in terms of number of vertices, over 57% of the Ark data with less than 20% of the load. The best performance is given by IPS with 58% of vertices and over 69% of edges. IPS was also able to send more probes before finishing the process, which is associated with the method's general performance and proves that rank ordering the monitor list avoids early termination. From the figure we can also see that there is some relation between the number of probes sent and the number of unique vertices and edges found.

MAX's under-performance compared to VPS might be due to the large /8 prefix range as logical landmark. Different addresses within the landmark might provide different sensitivities as different prefix sizes. It is also important to mention that MAX's built-in randomness produced by asynchronous probe arrival induces different results for every cycle. Since networks that belong to an equal /8 prefix use the same rank order list, their varying probes' arrival times might differ from the assigned monitor from cycle to cycle.

IPS has a lower degree of a random component, which is also due to asynchronous probe arrivals, but since monitor lists are specifically designed for each advertised prefix, randomness appears to have less impact on IPS.

We wanted to determine what improvement there is in the results if we increase the load of sent probes in SCP. We experimented with slightly modifying IPS's algorithm by incrementing the number of parent traces sent. In Chapter 4, we mentioned that IPS's strategy begins by finding ingress routers to the /8 prefix of traces directed to the target network. Then, the process follows by expanding the address range in order to find paths to the destination AS from every available monitor. We have taken the number of ingress routers before commencing the expansion procedure and approximated it to the smallest following integer power of two. That number might vary for every prefix, and two is the minimum number of parent traces sent. We have called this boosted version of IPS, IPS++. The results of this method can be seen in Figure 5.4, where just by increasing the load to 31%, we have incremented the vertices found to 80% and discovered 4% more edges than what Ark's methodology did. These results show that in the current stage of development of SCP, incrementing the probing load intelligently can even influence results more than monitor assignment techniques.

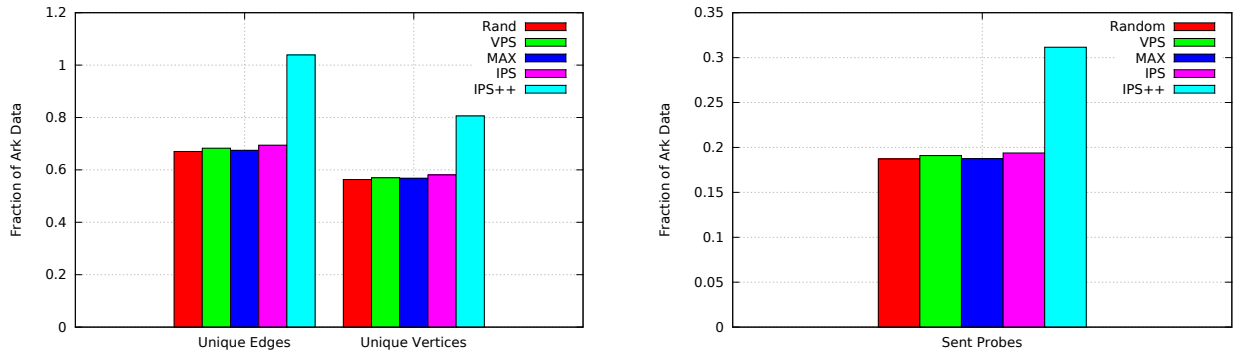

Figure 5.4. Results comparison between different methods.

In Figure 5.5 we depict the average number of vertices and edges found by each method. This metric helps to analyze the effectiveness of algorithms in terms of sent probes versus information learned. IPS++ has given a lower result compared to other intelligent monitor assigning methods, which was expected due to the probing load increment, but the method has learned over one vertex per sent probe.

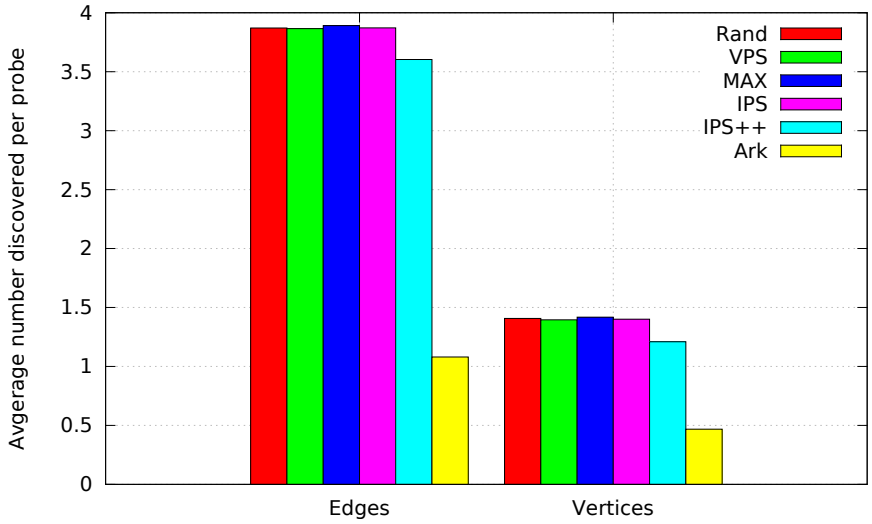

Figure 5.5. Average number of vertices and edges discovered per probe sent.

SCP is a topology discovery strategy that focuses on extracting networks' internal structures. We have illustrated in Figure 5.6 the fraction of discovered ingress routers and vertices inside the destination AS for all studied methods. Without considering IPS++, IPS is, as expected, the algorithm that captures the most ingress routers and inside target AS hops with a 6% and 14% over the total number of vertices found, respectively.

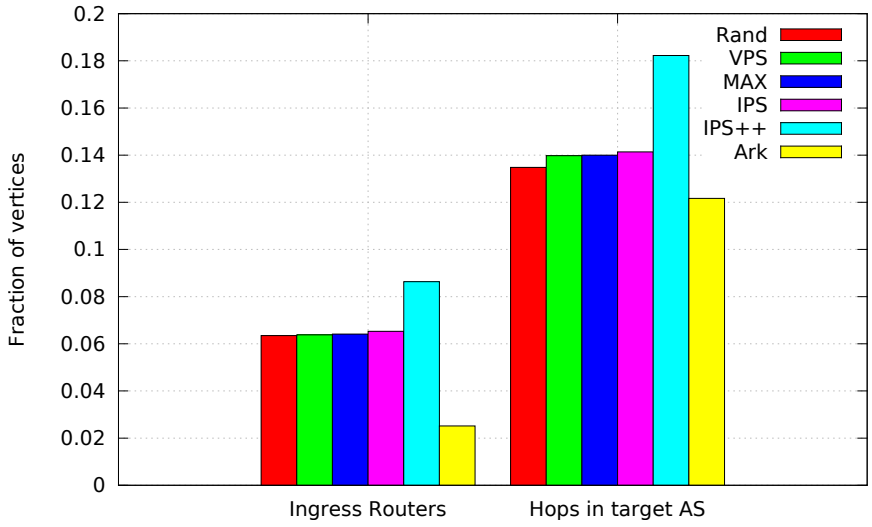

Figure 5.6. Fraction of vertices corresponding to ingress routers and hops inside target AS.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusions and Recommendations

In this thesis we designed and implemented promising probing strategies for network topology mapping. The primary objective has been achieved: successfully enabling SCP to conduct topology discovery over a subsection of the Internet and overcoming the challenges of implementing SCP amid Internet load-balancing. Additionally, SCP and VPS have been merged into one single algorithm, and we have developed new techniques that benefit SCP's performance.

A significant contribution toward SCP is the development of NID as a stopping criterion. NID has been shown to more effectively operate on the Internet, where load-balancing is prevalent, as compared to LD. This simple technique, besides focusing directly on the target networks' internal structure, has given SCP new capabilities, such as pair-wise probing independence. SCP can now assign disparate monitors to source each probe, even to the same target network, thereby increasing the opportunity to gather new interfaces, decrease the possibility of early termination and further remove the requirement to send two probes per prefix or sub-prefix.

After decoupling monitor assignment from SCP, the performance of various assignment methods were assessed. The results show that intelligently selecting monitors has benefit in comparison to existing random methods. We follow a strategy of analyzing prior rounds of probing in order to predict future probe behavior and better assign monitors. Among methods to use pre-probing data, we find that IPS produces the best results. By integrating IPS with SCP and utilizing known ingress routers, SCP is able to exploit path diversity. Further, by selecting monitors intelligently, SCP avoids early termination by ensuring that all known disjoint ingresses into the target network are traversed. The resulting performance is shown to increase the number of discovered vertices and edges as compared to random monitor selection.

While our results are promising, there is significant opportunity for further experiments, development and analysis. For example, additional study is warranted to assess whether the performance we achieve holds true in general. We leave larger-scale and Internet-scale studies for future work. Similarly, the results we present are macro, and do not examine how well we map particular networks. A per-network or per-prefix analysis could determine whether techniques perform better when mapping particular prefixes sizes, e.g., the performance of SCP when mapping small versus large prefixes. A per-prefix analysis of our results is left pending for future work.

Every comparison we have presented in this work has been done in terms of numbers of vertices and edges. A more thorough comparison of our methods and Ark's strategy, focusing in determining which specific interfaces are found by each method, is left for future work. For example, we wish to ascertain whether we find the same set of nodes as Ark and explore what the set difference reveals.

While SCP shows promise, its application to some known networks has revealed weaknesses. We have implemented our strategies over a university network announcing three /16 prefixes where we know much of the university's ground-truth. Unfortunately, our method does not capture any of this university's internal network structure as a result of premature termination. SCP sent two parent traces per prefix, yet none returned any hops inside the destination university ASes. However, manual inspection and manual traces to known web-servers of the university revealed routers and infrastructure that SCP missed. The root cause for the discrepancy lies with LCP where the chosen destination addresses result in probes without any hops inside the destination AS. Such behavior is rare, and our supposition is that the non-responsive hops are due to a particular firewall or policy configuration. Therefore, these results show that LCP has certain weak points that need to be studied. Because of our modular design, future work can test other destination selection methods, such as the nested IP block technique.

In this thesis, we have focused on topological coverage and probing savings rather than examining time savings. We expect time savings and the ability to reduce the probing cycle time to be correlated with the number of probes issued. Our probing restricted the maximum number of probes in flight; future work should examine the practical speed benefit of our algorithms. Considering that IPS++ sent only 30% of the probes as compared to Ark, we expect a significant time savings.

Longer term, SCP could run continuously to facilitate longitudinal analysis of the Internet topology. Difficulties arise in full-time probing when a monitor goes off-line, in particular for rank order lists where it is not as simple as just choosing the next monitor in the list. For instance, in IPS if two monitors, A and B, use the same ingress router towards a specific prefix, depending on the selected heuristic, one of them (in this example, B) has to go to the end of the list. Then, if monitor A goes off-line, B should replace it rather than the monitor following A in the list. Replacing off-line monitors can get even more complex when using MAX because of its path diversity algorithm. Therefore, several impediments to deploying SCP in production remain and must be properly analyzed in future work.

Several ideas have appeared during the discussions presented in this thesis, allowing us to go even deeper in our understanding of the proposed techniques. Further validation of this work could be performed by evaluating its results over ground truth. Diverse sources of data yield substantially different views of the Internet. Therefore, an analysis of the resulting differences between the extraction of prefixes from RIS of RIPE NCC is convenient. So far NID has been tested by counting new interfaces discovered. Different results might be found by including newly discovered edges in the metric. Evaluating MAX's sensitivity to different landmark sizes and different addresses within the landmarks might help to better understand MAX's behavior and potential. As we found also, IPS++ provided a substantial gain of information by intelligently increasing the probing load. Further work could test and compare the results of using different number of parent traces or adapting the number of sent probes, depending on feedback.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix: Program Files

SCP has been written using Python 2.7 programming language and implemented on the BSD operating system. SCP consists of four files: scp.py, arkmonitor.py, prefix.py and tod.py. The program is run by executing scp.py with the following command:

```
python scp.py -o [output] -f [traces_in_flight] -m [VP_assignment_method]
```

scp.py

```
1  import getopt
2  import sys
3  import time
4  import os
5  from random import shuffle
6
7  import arkmonitor
8  import bgpquery
9  import tod
10 from prefix import *
11
12 # Function that count new hops found in trace.
13 def cntNewInt(prefixhops, traceASNhops, parent = False):
14     new_hops = [x for x in traceASNhops if x not in prefixhops]
15
16     if parent: #for parent trace, all hops within  AS are counted
17         dist = len(traceASNhops)
18     else:
19         dist = len(new_hops)
20     print "Number of new Interfaces found:", dist
21     return dist, new_hops #dist is the number of new hops.
22                           #new_hops is added to universe of
23                           #known nodes inside dest AS.
24
25 def load_monitors(monfile):
26     monitors = {}
27     fl1 = open(monfile, 'r')
28     for line in fl1:
29         if len(line) > 2:
30             line = line[:-1]
31             ln = line.split(': ')
32             list1 = ln[1].split(',')
33             monitors[ln[0]] = list1
```

```
34      return monitors
35
36  def seedPrefixes(prefixes, count=10):
37      # Initial method to grab a set of BGP seed prefixes
38      b = bgpquery.BGPquery("localhost", 2002)
39      b.connect()
40      prefix_dict = {}
41
42      #Builds Set of available /8 prefixes in MAX lists.
43      maxfile = open('monitor_clas_max','r')
44      max_set = set()
45      for line in maxfile:
46        if len(line) < 2:
47          continue
48        ln = line.split(': ')
49        max_set.add(ln[0])
50
51      while len(prefix_dict) < count:
52          nets = []
53          (ip, mask, asn) = b.rand()
54
55          #Verifies available /8 prefixes MAX Set, so that it is
56          #possible to compare final results with different methods.
57          slash8 = arkmonitor.ArkMonitor().NetworkAddress(ip, 8)
58          if slash8 not in max_set:
59            continue
60
61          if int(mask) > 15 and int(mask) < 21:
62              net_list = []
63              for i in range(16,int(mask)+1):
64                  net = arkmonitor.ArkMonitor.NetworkAddress(ip,i)
65                  net_list.append(net)
66              nets = [x for x in net_list if x in prefix_dict]
67          if len(nets) > 1 and int(mask) >= 16 and
68             int(mask) < int(prefix_dict[nets[0]]):
69              p = Prefix(ip, prefix_dict[nets[0]], ip)
70          elif int(mask) >= 16:
71              prefix_dict[ip] = mask
72      for prefix in prefix_dict:
73          p = Prefix(prefix, prefix_dict[prefix], prefix)
74          p.parentmask = prefix_dict[prefix]
75          p.parentprefix = True
76          prefixes.append(p)
77      del b
78
```

```
79  def writePrefixes(prefixes, outfile):
80      try:
81          file = open(outfile, "w")
82      except IOError, err:
83          print err
84          sys.exit(0)
85      for prefix in prefixes:
86          file.write(prefix.id + "\n")
87
88  def submit(x):
89      print "Starting:", x
90
91  def finish(trace):
92      global file
93      global dst_to_prefix
94      global mon
95      global monitor_index
96      global ingress_routers
97      global identifier
98      EDthresh = 1
99
100     # have data to read.  fetch it and write it.
101     destASNhops, ingress = trace.hopASN()
102     prefix = dst_to_prefix[trace.dst]
103     prefix.probes -= 1
104
105     if ingress != '':
106         #assumes first hop as ingress router
107         ingress_routers[ingress] = prefix.parent
108
109     print "Got trace result for:"
110     prefix.dump()
111     status = trace.completeStatus(prefix)
112     print "Status:", status
113     print "Hops in AS: ", len(destASNhops), "Dest: ", trace.dst
114
115     if prefix.parent not in prefix_hops:
116         prefix_hops[prefix.parent] = []
117
118     if len(destASNhops)>0:
119         #in case destination decreased TTL before replying,
120         #therefore dest hop is included
121         if trace.dst == destASNhops[-1]:
122             #in trace, which is not used by the metric.
123             destASNhops = destASNhops[:-1]
```

```
124             trace.reply = 'R'
125
126     # Calculates the number of new interfaces;
127     # does not include destination IPs into count.
128     pre = prefix_hops[prefix.parent]
129     if prefix.parentprefix:
130         dist, new_hops = cntNewInt(pre, destASNhops, True)
131     else:
132         dist, new_hops = cntNewInt(pre, destASNhops)
133
134     if trace.reply == 'R': # checks whether destination replied
135         prefix_hops[prefix.parent] += [trace.dst]
136     prefix_hops[prefix.parent] += new_hops
137
138     if prefix.mask+1 == 33:
139         #Avoids dividing prefix into a /33 (invalid in IPv4)
140         pass
141     elif dist >= EDthresh:
142         # each prefix.toTargets creates two new targets
143         # (at .25 and .75)
144         print "Splitting."
145         (p1, p2) = Prefix.split(prefix)
146         t = trace.dst
147         p = prefix.mask + 1
148         i = identifier
149         mi = monitor_index
150         net = arkmonitor.ArkMonitor.NetworkAddress(t,p)
151         if mon:
152           if net == p1.ip:
153             identifier = p1.toTargets(targets,i,dst_to_prefix,mi)
154           else:
155             identifier = p2.toTargets(targets,i,dst_to_prefix,mi)
156         el:
157           if net == p1.ip:
158             identifier = p1.toTargets(targets,i,dst_to_prefix,mon)
159           else:
160             identifier = p2.toTargets(targets,i,dst_to_prefix,mon)
161
162
163 ###################################################################
164
165 try:
166     opts, args = getopt.getopt(sys.argv[1:], "hf:o:t:m:",
167         ["help", "out=", "flight=", "thresh=", "monitors="])
168 except getopt.GetoptError, err:
```

```
169     print str(err)
170     sys.exit(2)
171 
172 traces_in_flight = 2
173 mon = False # Default value
174 extended = False # Default value
175 for o, a in opts:
176     if o in ("-f", "--flight"):
177         traces_in_flight = int(a)
178     elif o in ("-t", "--thresh"):
179         EDthresh = int(a)
180     elif o in ("-o", "--out"):
181         outfile = a
182     elif o in ("-h", "--help"):
183         print sys.argv[0],"[-o][-f traces in flight][-t EDthresh]"
184         sys.exit()
185     elif o in ("-m", "--monitors"):
186         if a!='random' and a!='ips' and a!='max' and a!='vps':
187             print 'options are random, ips, max or vps only'
188             sys.exit()
189         else:
190             if a == 'random':
191                 mon = False
192             elif a == 'max' or a == 'ips' or a == 'vps':
193                 mon = a
194     elif o in ("-e", "--extended"):
195         extended = True
196     else:
197         assert False, "unhandled option"
198 
199 # Open ToD output
200 try:
201     file = open(outfile, "w")
202 except NameError or IOError, err:
203     print err, ": Must specify output file."
204     sys.exit(0)
205 
206 prefixes = []
207 # Gather seed prefixes
208 if 0:
209     seedPrefixes(prefixes, count=500)
210     writePrefixes(prefixes, "seed_prefixes")
211 elif 1:
212     seed = open('seed_prefixes', 'r')
213     for line in seed:
```

```
214         if len(line) > 1:
215             line = line[:-1].split(',')
216             p = Prefix(line[0], line[1], line[0])
217             p.parentmask = line[1]
218             p.parentprefix = True
219             prefixes.append(p)
220
221 else:
222     d5 = ('128.210.0.0', 16)
223     d6 = ('128.211.0.0', 16)
224     d7 = ('128.10.0.0', 16)
225     pref = [d5, d6, d7]
226     for pf in pref:
227         p = Prefix(pf[0], pf[1], pf[0])
228         p.parentmask = pf[1]
229         p.parentprefix = True
230         prefixes.append(p)
231
232 # Generate initial set of targets = (monitor, dst)
233 debug_lst = []
234 targets = []
235 dst_to_prefix = {}
236 identifier = 1
237
238 if mon == 'ips':
239   monitor_index = {}
240   monitors = load_monitors('monitor_clas_ips')
241   extended = True
242   i = 1
243   for pre in prefixes:
244     print i,
245     pre.dump()
246     net = arkmonitor.ArkMonitor.NetworkAddress(pre.ip, 8)
247     monitor_index[pre.ip] = 0
248     lista = monitors['/'.join([pre.ip,str(pre.mask)])]
249     indegree = lista.pop(0)
250     pre.mon_order = lista
251     #checks whether ips++ has been chosen or not.
252     if not(extended):
253         indegree = 0
254     iden = identifier
255     mi = monitor_index
256     ind = int(indegree)
257     identifier=pre.toTargets(targets,iden,dst_to_prefix,mi,ind)
258     i += 1
```

```
259
260 elif mon == 'max':
261   monitor_index = {}
262   monitor_index['max'] = 0
263   monitors = load_monitors('monitor_clas_max')
264   i = 1
265   for pre in prefixes:
266     print i,
267     pre.dump()
268     ip = pre.ip
269     # MAX establishes its /8 prefix as its parent prefix for
270     # monitor assignment
271     pre.parent=arkmonitor.ArkMonitor().NetworkAddress(ip,8)
272     monitor_index[pre.parent] = 0
273     pre.mon_order = monitors[pre.parent]
274
275     if extended:
276       indegree = 4 #set as default
277     else:
278       indegree = False
279
280     iden = identifier
281     mi = monitor_index
282     ind = int(indegree)
283     identifier=pre.toTargets(targets,iden,dst_to_prefix,mi,ind)
284     i += 1
285
286 #In VPS, each prefix has its own list.
287 #each list is randomly shuffled for monitor diversity.
288 elif mon == 'vps':
289   monitor_index = {}
290   monitors_list = arkmonitor.ArkMonitor().monitor_list
291   i = 1
292   for pre in prefixes:
293     print i,
294     pre.dump()
295     monitor_index[pre.ip] = 0
296     shuffle(monitors_list)
297     pre.mon_order = monitors_list
298
299     iden = identifier
300     mi = monitor_index
301     identifier=pre.toTargets(targets,iden,dst_to_prefix,mi)
302     i += 1
303
```

```
else:
  monitor_index = False
  i = 1
  for pre in prefixes:
    #if pre.ip != '79.170.190.0':
    #  continue
    if extended:
      indegree = 10
    else:
      indegree = False
    print i,
    i += 1
    pre.dump()

    iden = identifier
    mi = monitor_index
    ind = int(indegree)
    identifier=pre.toTargets(targets,iden,dst_to_prefix,mi,ind)

prefix_hops = {}
ingress_routers = {}
start = time.strftime('%X %x %Z')

arkmonitor.ArkMonitor.probe(submit,finish,targets,
                            file,traces_in_flight)
print "Start:", start
print "End:", time.strftime('%X %x %Z')
print "Total number of prefixes:", len(prefixes)
print "Threshold used:"
print "Traces in flight:", traces_in_flight
print "Ingress routers:", len(ingress_routers)
```

arkmonitor.py

```
import struct
import socket
import random
import subprocess
import select
import datetime

import tod

class ArkMonitor:
  def __init__(self, useBad=False):
```

```
12      #monitors.txt contains monitors names and IPs
13      f = open('monitors.txt', "r")
14      self.monitors = {}
15      for line in f:
16          if len(line) < 2:
17              continue
18          lst = line.split(': ')
19          lst[1] = lst[1][:-1]
20          self.monitors[lst[0]] = lst[1]
21
22      self.down_monitors = []
23  #=============================================================
24
25      if useBad == False:
26          for down in self.down_monitors:
27              del self.monitors[down]
28      moni = self.monitors
29      self.monitors_by_ip=dict((v,k) for k,v in moni.iteritems())
30      self.monitor_list = self.monitors.keys()
31      self.last_monitor = len(self.monitor_list) - 1
32
33    def getMonitors(self):
34        return self.monitor_list
35
36    def getRandMonitor(self):
37        monitor_index = random.randint(0,len(self.monitor_list)-1)
38        return self.monitor_list[monitor_index]
39
40    def get(self, rand=False):
41        if rand:
42            return self.getRandMonitor()
43        else:
44            return self.getNextMonitor()
45
46    @staticmethod
47    def dottedQuadToNum(ip):
48        "convert decimal dotted quad string to long integer"
49        return struct.unpack('!I',socket.inet_aton(ip))[0]
50
51    @staticmethod
52    def numToDottedQuad(n):
53        "convert long int to dotted quad string"
54        return socket.inet_ntoa(struct.pack('!I',n))
55
56    @staticmethod
```

```
57    def NetworkAddress(ip, bits):
58        ipaddr = ArkMonitor.dottedQuadToNum(ip)
59        mask = (0xffffffff << (32 - int(bits))) & 0xffffffff
60        n = ipaddr & mask
61        netaddr = ArkMonitor.numToDottedQuad(n)
62        return netaddr
63
64    # Talk to Ark
65    @staticmethod
66    def probe(submit_hook,finish_hook,targets,file,traces_in_flight=2):
67      # popen() Ark ToD service
68      print "Interacting with Ark ToD, will maintain",
69            traces_in_flight, "traces in flight."
70      proc = subprocess.Popen(['./tod-client','--session-id=cmand'],
71                      shell=False,
72                      stdin=subprocess.PIPE,
73                      stdout=subprocess.PIPE,)
74
75      time_control = datetime.datetime.now()
76      non_responsive = 0 # counts non-responsive traces
77      expired = 0 # expired traces
78      sent_probes = {}
79      rcvd_probes = {}
80      traces_active = {}
81      delay = 30 # minutes
82      fails = open('non_responsive','w')
83      while len(targets) > 0 or len(traces_active) > 0:
84        print "Traces active:", len(traces_active),
85              "Targets remaining:", len(targets)
86
87 ##################################################################
88        # spawn a new ToD trace if too few are in flight and
89        # there are more to add
90        while (len(traces_active) < traces_in_flight) and
91            (len(targets) > 0):
92          tgt = targets.pop(0)
93          tgtx = tgt.split(' ')
94          traces_active[tgtx[0]]=[datetime.datetime.now(),
95                                  tgtx[1],tgtx[3]]
96          proc.stdin.write(tgt)
97          proc.stdin.write("\n")
98          sent_probes[tgtx[0]] = (tgtx[1],tgtx[3])
99          submit_hook(tgt)
100
101 ##################################################################
```

```
102       diff = datetime.datetime.now() - time_control
103       if diff = datetime.timedelta(minutes=(delay-1)):
104         time_control = datetime.datetime.now()
105
106         #Searches for traces that have gone over the delay,
107         #in cycles equal to the delay (to reduce overhead).
108         for trace in traces_active.keys():
109           ta = traces_active[trace][0]
110           timedelta = datetime.datetime.now() - ta
111           if timedelta >= datetime.timedelta(minutes=delay):
112             del traces_active[trace]
113             expired += 1
114
115 ###############################################################
116       # wait for data from tod-bulk-probe
117       fdready = select.select([proc.stdout], [], [], 10)
118
119       # Repeats the cycle if no probe has been received
120       if len(fdready[0]) == 0:
121           continue
122
123       # have data to read.  fetch it and process
124       out = proc.stdout.readline()
125
126       trc = tod.ToD(out)
127       rcvd_probes[trc.id]=(ArkMonitor().monitors_by_ip[trc.src],trc.dst)
128       trc.writeout(file)
129
130       finish_hook(trc)
131       if trc.id in traces_active:
132           del traces_active[trc.id]
133       else:
134           expired -=1
135     for trace_id in sent_probes:
136         if trace_id not in rcvd_probes:
137             line=str(trace_id)+','.join(sent_probes[trace_id])+'\n'
138             fails.write(line)
139             non_responsive += 1
140
141     print "Number of expired traces",expired
142     print 'Number of probes sent:',len(sent_probes)
143     print 'Number of probes received:',len(rcvd_probes)
144     print 'Number of non_responsive probes:',non_responsive
```

prefix.py

```
import math

from arkmonitor import *

class Prefix:
  def __init__(self, _ip, _mask, _parent = False):
    self.ip = _ip            # prefix's ip address
    self.mask = int(_mask)
    self.id = ",".join([self.ip, str(self.mask)])
    self.parent = _parent # prefix of its parent prefix
    self.parentmask = 0
    self.parentprefix = False
    self.probes = 0
    self.hops = []
    self.mon_order = [] # holds monitor rank order list.

  def dump(self):
    print "Prefix:", self.ip, "/", self.mask

  def contains(self, ip):
    ip1 = ArkMonitor.dottedQuadToNum(self.ip)
    ip2 = ArkMonitor.dottedQuadToNum(ip)
    ones = 0xffffffff
    mask = ones & (ones<<int(self.mask))
    if (ip1 & mask) == (ip2 & mask):
        return True
    return False

  # take a prefix and turn into two targets for Ark.
  def toTargets(self, targets, identifier, dst_to_prefix,
                monitor_index = False, indegree = False):
    if indegree == False or indegree < 3:
        indegree = 2
        indegree_pow2 = indegree
        power = 1
    else:
        ceil1 = math.log(indegree,2)
        power = math.ceil(ceil1)
        indegree_pow2 = 2**power
        if indegree_pow2 > len(self.mon_order) and
           len(self.mon_order) > 0:
          while indegree_pow2 > len(self.mon_order):
              indegree -= 1
```

```
44                power = math.ceil(math.log(indegree,2))
45                indegree_pow2 = 2**power
46
47      base = ArkMonitor.dottedQuadToNum(self.ip)
48      size = 2**(32-int(self.mask))
49
50      for i in range(int(indegree_pow2)):
51        if self.parentprefix:
52          newsize = 2**(32-int(self.mask)-power)
53          middle = base + i*newsize
54          new_net = ArkMonitor.numToDottedQuad(middle)
55          pref = Prefix(new_net, self.mask + power - 1, self.parent)
56          pref.parentprefix = True
57        else:
58          pref = Prefix(self.ip, self.mask, self.parent)
59        t=int(base+size*(1.0/(2*indegree_pow2)+1.0*i/indegree_pow2)-2)
60        dst = ArkMonitor.numToDottedQuad(t)
61
62        if not monitor_index:
63          mon = ArkMonitor().get(rand=True)
64        else:
65          index = monitor_index[self.parent] % len(self.mon_order)
66          monitor_index[self.parent] += 1
67
68          mon = self.mon_order[index]
69          pref.mon_order = self.mon_order
70          pref.parentmask = self.parentmask
71
72        pref.probes = indegree_pow2
73        targets.append(str(identifier)+" "+mon+" trace "+dst)
74        identifier += 1
75        dst_to_prefix[dst] = pref
76      return identifier
77
78    def addHops(self, hops):
79      for i in range(len(hops)):
80        hop = hops[i]
81        if len(self.hops) <= i:
82            self.hops.append({})
83        self.hops[i][hop] = 1
84        num = len(self.hops[i])
85        print num, "Interfaces seen at hop:", i + 1
86        for j in self.hops[i]:
87            print "\t", j
88
```

```
89    @staticmethod
90    def split(prefix):
91      mask = prefix.mask + 1
92      m = Prefix.middle(prefix)
93      p1 = Prefix(prefix.ip, mask, prefix.parent)
94      p1.mon_order = prefix.mon_order
95      p1.parentmask = prefix.parentmask
96      p2 = Prefix(m, mask, prefix.parent)
97      p2.mon_order = prefix.mon_order
98      p2.parentmask = prefix.parentmask
99      return (p1, p2)
100
101   @staticmethod
102   def middle(prefix):
103       base = ArkMonitor.dottedQuadToNum(prefix.ip)
104       halfsize = 2**(32-int(prefix.mask)-1)
105       middle = base + halfsize
106       return ArkMonitor.numToDottedQuad(middle)
107
108 def index(string, list_length):
109     suma = 0
110     for n in range(0, len(string)):
111         suma += 2**n
112     suma += int(string, 2) - 1
113     return suma % list_length
```

tod.py

```
1  import bgpquery
2
3  class ToD:
4    def __init__(self, line):
5      self.line = line
6      self.range = [0,0]
7      self.used = False
8      self.reached_prefix = False
9      fields = line.rstrip().split("\t")
10     self.id = fields[0].split(' ')[0]
11     (self.src, self.dst)    = fields[1:3]
12     (self.cycle, self.ts)   = fields[4:6]
13     (self.reply, self.rtt)  = fields[6:8]
14     (self.ttl, self.rttl)   = fields[8:10]
15     (self.stat, self.halt)  = fields[10:12]
16     self.comp  = fields[12]
17     self.ASN = 0
```

```
18      self.ASNhops = []
19      self.destASNhops = [] #Gathers interfaces that belong to
20                            #the destination ASN.
21      self.hops = []
22      for hop in fields[13:]:
23          if hop[0] == 'q':
24              self.hops.append('0.0.0.0')
25              continue
26          if hop.find(";") > -1:
27              hop = hop.split(";")[0]
28          (ip, ttl, tries) = hop.split(",")
29          self.hops.append(ip)
30
31  def printSummary(self):
32      print "Trace:", self.src, "->", self.dst, "[",
33             self.range[0], ":", self.range[1], "]"
34
35  def coversize(self):
36      if self.range[0] == 0 and self.range[1] == 0:
37          return 0
38      return self.range[1] - self.range[0] + 1
39
40  def addcoverhop(self, hop_num):
41      if self.range[0] == 0:
42          self.range[0] = hop_num
43      self.range[1] = hop_num
44
45  def reset(self):
46      self.range = [0,0]
47
48  def writeout(self, file):
49      file.write(self.line)
50
51  def show(self, start=1, end=9999):
52      print "Trace:", self.src, "->", self.dst,
53            "ASN:", self.ASN,
54      if start != 1 or end != 9999:
55          print "range:", start, "->", end,
56      if end > len(self.hops):
57          end = len(self.hops)
58      print
59      for i in range(start, end + 1):
60          print "\t", i, "Hop:", self.hops[i-1]
61
62  def store(self, start=1, end=9999):
```

```
63      text =  "Trace:"+str(self.src)+"->"+str(self.dst)+\
64              "ASN:"+str(self.ASN)+"\n"
65      if start != 1 or end != 9999:
66        text = text+"range:"+str(start)+"->"+str(end)+"\n"
67      if end > len(self.hops):
68        end = len(self.hops)
69      for i in range(start, end + 1):
70        text = text+"\t"+str(i)+"Hop:"+str(self.hops[i-1])
71      return text
72
73    def getDst(self):
74        return self.dst
75
76    def add(self, interface_dict):
77        for hop in self.hops:
78            if interface_dict.has_key(hop):
79                interface_dict[hop]+=1
80            else:
81                interface_dict[hop]=1
82
83    def differAt(self, trace):
84        s = self.hops
85        t = trace.hops
86        diff = []
87        l = len(s)
88        if (len(t) < l):
89            l = len(t)
90        for i in range(l):
91            if s[i] == '0.0.0.0' or t[i] == '0.0.0.0':
92                continue
93            if s[i] != t[i]:
94                diff.append(i)
95        return diff
96
97    def completeStatus(self, prefix):
98        for hop in self.hops:
99            if prefix.contains(hop):
100               self.reached_prefix = True
101               return True
102       return False
103
104   def hopASN(self):
105     min_asn_hops = 0
106     dASNhops = []
107     last_hop = ''
```

```
108     ingress = ''
109     b = bgpquery.BGPquery("localhost", 2002)
110     b.connect()
111     (ip, mask, asn) = b.lookup(self.dst)
112     self.ASN = asn
113     for hop in self.hops:
114       (ip, mask, asn) = b.lookup(hop)
115       self.ASNhops.append(asn)
116       # checks whether ip belongs to destination ASN
117       if asn == self.ASN:
118         dASNhops.append(hop)
119         if ingress == '' and '*' not in last_hop:
120           ingress = last_hop
121       last_hop = hop # holds last hop outside the dest. AS.
122     l = len(dASNhops)
123     if l < min_asn_hops: # at least 1 hop in dASNhops
124         xhops = self.hops[-(min_asn_hops+1)-l:-l-1]
125         dASNhops = xhops + dASNhops
126     del b
127     return dASNhops, ingress
```

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] S. Radicati and Q. Hoang. (2011, Nov.) Email statistics report, 2011-2015. Executive Summary. The Radicati Group. [Online]. Available: http://www.radicati.com/wp/wp-content/uploads/2011/05/Email-Statistics-Report-2011-2015-Executive-Summary.pdf

[2] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: from art to science," in *IEEE Conference For Homeland Security, Cybersecurity Applications & Technology*, pp. 205–211, 2009.

[3] X. Lang, G. Zhou, C. Gong, and W. Han, "Dolphin: The measurement system for the next generation Internet," in *4th International Conference on Communications, Internet and Information Technology*, pp. 1–12, 2005.

[4] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *7th Symposium on Operating Systems Design and Implementation*, pp. 367–380, 2006.

[5] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 71–74, 2005.

[6] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[7] M. H. Gunes and K. Sarac, "Resolving anonymous routers in Internet topology measurement studies," in *27th Conference on Computer Communications*. IEEE, pp. 1076–1084, 2008.

[8] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 327–338, 2005.

[9] H. Kardes, M. Gunes, and T. Oz, "Cheleby: A subnet-level Internet topology mapping system," in *IEEE 4th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–10, 2012.

[10] M. Chen, M. Xu, and K. Xu, "A delay-guiding source selection method in network topology discovery," in *IEEE International Conference on Communications*, pp. 1–6, 2011.

[11] R. Beverly, A. Berger, and G. G. Xie, "Primitives for active Internet topology mapping: Toward high-frequency characterization," in *10th ACM SIGCOMM Conference on Internet Measurement*, pp. 165–171, 2010.

[12] D. Meyer. (2013, Jul.) Routeviews project. [Online]. Available: http://www.routeviews.org

[13] RIPE NCC. (2013, Jul.) Routing information service. [Online]. Available: http://www.ripe.net/data-tools/stats/ris/routing-information-service

[14] J. Postel, "RFC 760: DoD standard Internet protocol," in *Obsoleted by RFC0791, RFC0777 [27, 26], Obsoletes IEN123 [32]*, 1980.

[15] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," in *RFC 1771, Internet Engineering Task Force*, 1995.

[16] U. Weinsberg, Y. Shavitt, and E. Shir, "Near-deterministic inference of AS relationships," in *INFOCOM Workshops*. IEEE, pp. 1–2, 2009.

[17] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, A. Vahdat *et al.*, "The internet as-level topology: three data sources and one definitive metric," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 17–26, 2006.

[18] B. Zhang, R. Liu, D. Massey, and L. Zhang, "Collecting the Internet AS-level topology," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 1, pp. 53–61, 2005.

[19] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, pp. 346–353, 2001.

[20] J. Tomasik and M.-A. Weisser, "Internet topology on AS-level: model, generation methods and tool," in *IEEE 29th International Performance Computing and Communications Conference (IPCCC)*, pp. 263–270, 2010.

[21] M.-A. Weisser and J. Tomasik, "Automatic induction of inter-domain hierarchy in randomly generated network topologies," in *Proceedings of the Spring Simulation Multiconference*, vol. 1, pp. 77–84, 2007.

[22] H. Chang, S. Jamin, and W. Willinger, "To peer or not to peer: Modeling the evolution of the Internet's AS-level topology," in *IEEE International Conference on Computer Communications*, pp. 1–12, 2006.

[23] Y.-J. Chi, R. Oliveira, and L. Zhang, "Cyclops: the AS-level connectivity observatory," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 5–16, 2008.

[24] M. Lad, L. Zhang, and D. Massey, "Link-rank: A graphical tool for capturing bgp routing dynamics," in *Network Operations and Management Symposium*, vol. 1. IEEE, pp. 627–640, 2004.

[25] T. Bates, P. Smith, and G. Huston. (2013, Jul.) The CIDR report. [Online]. Available: http://www.cidr-report.org/as2.0

[26] X. Xu, Z. M. Mao, and J. A. Halderman, "Internet censorship in China: Where does the filtering occur?" in *Passive and Active Measurement*, pp. 133–142, 2011.

[27] K. Keys, "Internet-scale IP alias resolution techniques," vol. 40, no. 1, 2010, pp. 50–55.

[28] R. Siamwalla, R. Sharma, and S. Keshav, "Discovering Internet topology," in *Unpublished Manuscript*, 1998.

[29] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with paris traceroute," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pp. 153–158, 2006.

[30] B. Augustin, T. Friedman, and R. Teixeira, "Measuring load-balanced paths in the Internet," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pp. 149–160, 2007.

[31] J. Postel, M. Kosters, D. Karrenberg, and D. Conrad. (2013, Jul.) RFC2050: Internet registry IP allocation guidelines. IANA. [Online]. Available: http://tools.ietf.org/html/rfc2050

[32] (2013, Jul.) ICP-2: Criteria for establishment of new regional Internet registries. [Online]. Available: http://www.icann.org/en/resources/policy/global-addressing/new-rirs-criteria.2013

[33] ARIN. (1997, Dec.) Regional Internet Registries. [Online]. Available: http://www.arin.net/knowledge/rirs.html

[34] M. Fomenkov, E. Katz-Bassett, R. Beverly, B. A. Cox, M. Luckie *et al.*, "The workshop on active Internet measurements (AIMS) report," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 32–36, 2009.

[35] B. Huffaker, D. Plummer, D. Moore, and K. Claffy, "Topology discovery by active probing," in *Symposium on Applications and the Internet (SAINT) Workshops*. IEEE, pp. 90–96, 2002.

[36] M. Luckie, "Scamper: A scalable and extensible packet prober for active measurement of the Internet," in *10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 239–245.

[37] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," in *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[38] B. Huffaker, M. Fomenkov, and k. claffy, "Internet topology data comparison," in *Cooperative Association for Internet Data Analysis (CAIDA)*, 2012.

[39] Y. Tian, R. Dey, Y. Liu, and K. W. Ross, "China's Internet: Topology mapping and geolocating," in *INFOCOM*. IEEE, 2012, pp. 2531–2535.

[40] T. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *IEEE INFOCOM*, vol. 1, pp. 170–179, 2002.

[41] X. Zou, Z. Qiao, G. Zhou, and K. Xu, "A logic distance-based method for deploying probing sources in the topology discovery," in *IEEE Global Telecommunications Conference*, pp. 1–6, 2009.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California